# Kimball Data Warehouse Toolkit

## Ch 1 - Data Warehousing, Business Intelligence, and Dimensional Modeling Primer

### Intro

* The data warehousing and business intelligence (DW/BI) industry has since Kimball published the 1st edition of *The Data Warehouse Toolkit* (Wiley) in 1996
* Although large corporate early adopters paved the way, DW/BI has since been embraced by organizations of all sizes + the industry has built thousands of DW/BI systems
* The volume of data continues to grow as DW’s are populated with increasingly **atomic data** and updated with greater frequency
* Databases have grown from MB to GB to TB to PB, yet the basic challenge of DW/BI systems has remained remarkably constant
* Job = to marshal an organization’s data + bring it to business users for their decision making
* Collectively, business professionals everywhere are making better decisions + generating payback on their DW/BI investments.
* Since the 1st edition of *Toolkit* was published, **dimensional modeling** has been broadly accepted as the dominant technique for DW/BI presentation
* Practitioners and pundits alike have recognized that the **presentation of data must be grounded in simplicity** if it is to stand any chance of success
* Simplicity is the fundamental key that allows users to easily understand databases and software to efficiently navigate databases
* In many ways, **dimensional modeling** amounts to holding the fort against assaults on simplicity
* By consistently returning to a business-driven perspective and by refusing to compromise on the goals of **user understandability and query performance**, you establish a coherent design that serves the organization’s analytic needs
* This dimensionally modeled framework becomes the platform for BI
* Dimensional modeling is absolutely critical to a successful DW/BI initiative + also has emerged as the leading architecture for building integrated DW/BI systems
* When you use the conformed **dimensions** and conformed **facts** of a set of dimensional models, you have a **practical and predictable framework for incrementally building complex DW/BI systems that are inherently distributed**
* The core dimensional modeling techniques Kimball published 17 years ago have withstood the test of time
* Concepts such as **conformed dimensions**, **slowly changing dimensions**, **heterogeneous products**, **factless fact tables**, and the **enterprise data warehouse bus matrix** continue to be discussed in design workshops around the globe
* Original concepts have been embellished and enhanced by new and complementary techniques.
* 3rd edition = useful to summarize collective dimensional modeling experience under a single cover
* This book is loaded with specific, practical design recommendations based on real-world scenarios.
* Goal of this book = to provide a one-stop shop for dimensional modeling techniques
* It is a toolkit of dimensional design principles + techniques
* Addresses the needs of those just starting in dimensional DW/BI + describes advanced concepts for those of you who have been at this a while
* Even if not directly responsible for the dimensional model, it’s important for all members of a project team to be comfortable with dimensional modeling concepts
* **The dimensional model has an impact on most aspects of a DW/BI implementation**, beginning with the **translation of business requirements**, through the **ETL processes**, and finally, to the **unveiling of a DW through BI applications**
* Due to the broad implications, you need to be conversant in dimensional modeling regardless of whether you are responsible primarily for project management, business analysis, data architecture, database design, ETL, BI applications, or education and support
* Primarily discusses dimensional modeling in the context of a **relational database** (RDB) with nuances for **online analytical processing (OLAP) cubes** noted where appropriate
* First and foremost, the **DW/BI system must consider the needs of the business**
* With business needs firmly in hand, we work backwards through the logical and then physical designs, along with decisions about technology and tools
* We drive stakes in the ground regarding the goals of data warehousing and BI, while observing the uncanny similarities between the responsibilities of a DW/BI manager and those of a publisher

### Different Worlds of Data Capture and Data Analysis

* One of the **most important assets** of any organization is its **information**
* This asset is almost always used for 2 purposes: ***operational* record-keeping** and ***analytical* decision-making**
* Simply speaking, the **operational systems** = where you **put the data in**, + the **DW/BI system** = where you **get the data out**
* **Users of an operational system** **turn the wheels of the organization**
* Take orders, sign up new customers, monitor status of operational activities, + log complaints
* **Operational systems are optimized to *process transactions quickly***, almost always **deal with one transaction record at a time** +predictably **perform the same operational tasks over and over**, executing the organization’s business processes
* Given this execution focus, operational systems **typically do *not* maintain history**, but **rather** **update data to reflect the most current state**
* **Users of a DW/BI system** ***watch* the wheels of the organization** turn to **evaluate performance**
* Count new orders + compare w/ last week’s orders, ask why new customers signed up, what customers complained about, + worry about whether operational processes are working correctly
* Although DW/BI users **need detailed data to support their constantly changing questions**, they **almost *never* deal with one transaction at a time**
* **DW/BI systems are optimized for high-performance queries** as users’ questions often require that hundreds or hundreds of thousands of transactions be searched + compressed into an answer set
* To further complicate matters, **users of a DW/BI system typically demand that historical context be preserved to accurately evaluate the organization’s performance over time**
* The DW/BI system has profoundly different needs, clients, structures, + rhythms than the operational systems of record
* Unfortunately, we still encounter supposed DW/BI systems that are mere copies of the operational systems of record stored on a separate hardware platform
* Although these environments may address the need to isolate the operational and analytical environments for performance reasons, they do nothing to address the other inherent differences between the 1 types of systems
* **Business users are underwhelmed by the usability and performance provided by these *pseudo*-DW’s; these imposters do a disservice to DW/BI because they don’t acknowledge their users have drastically different needs than operational system users**

### Goals of Data Warehousing and BI

* Recurring themes have existed for more > 3 decades + are still so universal that they drive the bedrock requirements for the DW/BI system
* “We collect tons of data, but we can’t access it.”
* “We need to slice and dice the data every which way.”
* “Business people need to get at the data easily.”
* “Just show me what is important.”
* “We spend entire meetings arguing about who has the right numbers rather than making decisions.”
* “We want people to use information to support more fact-based decision-making.”
* We can turn these business management quotations into **requirements:**
* **The DW/BI system must make information easily accessible**
* i.e., *simple and fast*
* The **contents** of a DW/BI system must be **understandable**
* The **data** must be **intuitive + obvious** to the business user, not merely the developer
* The **data’s structures + labels should mimic the business users’ thought processes + vocabulary**
* Business users want to separate and combine analytic data in endless combinations, so **BI tools and applications that access the data must be simple and easy to use** + also must **return query results to the user with minimal wait times**
* **The DW/BI system must present information consistently**
* The **data** in a DW/BI system must be **credible** + must be **carefully assembled from a variety of sources, cleansed, quality assured, + released only when fit for user consumption**
* *Consistency* also implies **common labels and definitions** for the DW/BI system’s contents are **used across data sources**
* If 2 performance measures have the same name, they must mean the same thing
* Conversely, if 2 measures don’t mean the same thing, they should be labeled differently
* **The DW/BI system must adapt to change**
* User needs, business conditions, data, + tech are all subject to change
* A **DW/BI system must be designed to handle inevitable change gracefully** so that it doesn’t invalidate existing data or applications
* Existing data and applications should *not* be changed or disrupted when the business community asks new questions or new data is added to the DW
* Finally, if descriptive data in the DW/BI system must be modified, you must appropriately account for the changes and make these changes transparent to the users
* **The DW/BI system must present information in a timely way**
* As the DW/BI **system** is **used more intensively for operational decisions**, **raw data may need to be converted into actionable information within hours, minutes, or even seconds**
* The DW/BI team and business users need to have **realistic expectations** for what it means **to** **deliver data** when there is little time to clean or validate it
* **The DW/BI system must be a secure bastion that protects the information assets**
* An organization’s informational crown jewels are stored in the DW, potentially harmful details in the hands of the wrong people
* A DW/BI system must effectively control access to the organization’s confidential information
* **The DW/BI system must serve as the authoritative and trustworthy foundation for improved decision making**
* The DW **must have the *right* data to support decision making**
* **Most important output(s) from a DW/BI system = the decisions made based on analytic evidence presented that deliver the business impact + value attributable to a DW/BI system**
* Original label that predates DW/BI is still the best description of what to design: a ***decision support system***
* **The business community must *accept* the DW/BI system to deem it successful**
* It **doesn’t matter** that you built an elegant solution using best-of-breed products and platforms **if the business community does not embrace the DW/BI environment** and **actively use it**, because then you have failed the **acceptance test**
* Unlike an *operational* system implementation where business users *have no choice* but to use the new system, DW/BI usage is sometimes optional
* **Business users will embrace the DW/BI system if it is the “*simple and fast*” source for actionable information**
* Although each requirement on above is important, the final 2 (highlighted) are the most critical, and unfortunately, often the most overlooked
* **Successful data warehousing and BI demands more than being a stellar architect, technician, modeler, or DBA**
* With a DW/BI initiative, you have one foot in your **IT comfort zone** while the other foot is on the unfamiliar **turf of *business users***
* **Must straddle the two, modifying some tried-and-true skills to adapt to the unique demands of DW/BI**
* Need to bring a spectrum of skills to the party to behave like you’re a hybrid DBA/MBA

#### Publishing Metaphor for DW/BI Managers

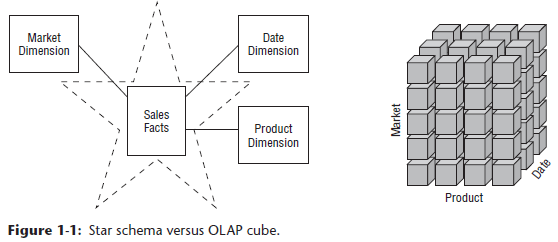
* Let’s compare the responsibilities of DW/BI managers with those of a publishing editor-in-chief (EIC)
* As the EIC of a high-quality magazine, you’d have broad latitude to manage the magazine’s content, style, + delivery
* Anyone with this EIC job title would likely tackle the following activities:
* **Understand the readers**:
* Identify their demographic characteristics.
* Find out what readers want in this kind of magazine.
* Identify the “best” readers who will renew their subscriptions + buy products from the magazine’s advertisers.
* Find potential new readers and make them aware of the magazine.
* **Ensure the magazine appeals to the readers:**
* Choose interesting and compelling magazine content.
* Make layout and rendering decisions that maximize the readers’ pleasure
* Uphold high-quality writing and editing standards while adopting a consistent presentation style
* Continuously monitor the accuracy of the articles and advertisers’
* claims.
* Adapt to changing reader profiles and the availability of new input from a network of writers and contributors.
* **Sustain the publication:**
* Attract advertisers and run the magazine profitably.
* Publish the magazine on a regular basis.
* Maintain the readers’ trust.
* Keep the business owners happy.
* We can also can identify items that should be ***non*-goals** for the EIC, such as building the magazine around a particular printing technology or exclusively putting management’s energy into operational efficiencies, such as imposing a technical writing style that readers don’t easily understand, or creating an intricate + crowded layout that is difficult to read
* By building the publishing business on a **foundation of serving the readers effectively**, the magazine is likely to be successful
* Conversely, go through the list + imagine what happens if you omit any single item; ultimately, the magazine would have serious problems
* There are strong parallels that can be drawn between being a conventional publisher vs. a DW/BI manager
* **Driven by the needs of the business, DW/BI managers must publish data that has been collected from a variety of sources + edited for quality and consistency** + the **main responsibility = to serve the business users**
* The publishing metaphor underscores the **need to focus outward on customers rather than merely focusing inward on products + processes**
* Although we use tech to deliver the DW/BI system, the **tech is at best a means to an end**
* As such, the **tech and techniques used to build the system should not appear directly in your top job responsibilities**
* Now recast the magazine publisher’s responsibilities as DW/BI manager responsibilities:
* **Understand the business users:**
* Understand their **job responsibilities, goals, + objectives**.
* **Determine** the **decisions** that business **users want to make** with the help of a DW/BI system
* **Identify the “best” users** who make effective, high-impact decisions.
* **Find potential new users and make them aware** of the DW/BI system’s capabilities
* **Deliver high-quality, relevant, and accessible information and analytics to the business users:**
* **Choose** the **most robust, actionable data to present** in the DW/BI system, carefully selected from the vast universe of possible data sources in the organization.
* **Make the UI’s and applications simple** and template-driven, explicitly matched to users’ cognitive processing profiles.
* **Make sure data is accurate + can be trusted**, **labeling** it **consistently** across the enterprise.
* **Continuously monitor the accuracy** of the data + analyses.
* **Adapt** to changing user profiles, requirements, + business priorities, along with the availability of new data sources.
* **Sustain the DW/BI environment:**
* **Take a portion of the credit for the business decisions made using the DW/BI** system + use these successes to **justify staffing and ongoing expenditures**
* **Update** the DW/BI **system on a regular basis**
* **Maintain** the business **users’ trust**
* Keep the business users, executive sponsors, + IT management happy
* If you do a good job with all these responsibilities, you will be a great DW/BI manager
* Conversely, go through the list and imagine what happens if you omit any single item (Ultimately, the environment would have serious problems)
* Now contrast this view of a DW/BI manager’s job with your *own* job description
* Chances are the preceding list is more oriented toward user + business issues and may not even sound like a job in IT (this is what makes data warehousing and BI interesting)

### Dimensional Modeling Introduction

* After understanding DW/BI system goals, **consider basics of dimensional modeling** **=****longstanding + widely accepted as the preferred technique for presenting analytic data/making databases simple because it addresses 2 simultaneous requirements**:
* **1) Deliver data that’s understandable to the business users**
* **2) Deliver fast query performance**
* In case after case, for > 5 decades, IT organizations, consultants, + business users have naturally gravitated to a simple dimensional structure to match the fundamental human need for **simplicity**
* **Simplicity is critical = it ensures that users can easily understand the data, as well as allows software to navigate and deliver results quickly and efficiently**
* Imagine an executive who describes her business as, “We sell products in various markets + measure our performance over time.”
* Dimensional designers listen carefully to the *emphasis on product, market, + time*
* Most people find it intuitive to think of such a business as a **cube** of data, with the edges labeled product, market, + time
* Imagine slicing and dicing along each of these dimensions
* **Points inside the cube are where the measurements, such as sales volume or profit, for that combination of product, market, and time are stored**
* **The ability to visualize something as abstract as a set of data in a concrete and tangible way is the secret of understandability**
* If this perspective seems too simple, good!
* **A data model that starts simple has a chance of remaining simple at the end of the design** + a **model that starts complicated surely will be overly complicated at the end**, resulting in slow query performance and business user rejection
* Although **dimensional models are often instantiated in RDBMSs, they are quite different from** **third normal form (3NF) models**which seek to **remove data redundancies**
* **Normalized 3NF structures divide data into many discrete entities, each of which becomes a relational table**
* A database of sales orders might start with a record for each order line but turn into a complex spider web diagram as a 3NF model, perhaps consisting of hundreds of normalized tables.
* The industry sometimes refers to 3NF models as **entity-relationship (ER) models**
* **Entity-relationship diagrams (ER diagrams or ERDs)** **= drawings that communicate the relationships between tables**
* ***Both* 3NF and dimensional models can be represented in ERDs because both consist of joined relational tables**
* The **key difference between 3NF and dimensional models** is the **degree of normalization**
* Because *both* model types can be presented as ERDs, **refrain from referring to 3NF models as ER models** + **instead,** **call them normalized models** to minimize confusion
* **Normalized 3NF structures** are **immensely useful in *operational* processing because an update or insert transaction touches the database in only one place**.
* **Normalized models, however, are too complicated for BI queries**
* Users can’t understand, navigate, or remember normalized models that resemble a map of the LA freeway system
* Likewise, **most RDBMSs can’t efficiently query a normalized model**, as the **complexity** of users’ **unpredictable queries overwhelms database optimizers**, resulting in **disastrous query performance**
* The **use of normalized modeling in the DW/BI presentation area defeats the intuitive and high-performance retrieval of data**
* Fortunately, **dimensional modeling addresses the problem of overly complex schemas in the presentation area**
* **NOTE:** A **dimensional model** contains the ***same information* as a normalized model**, but **packages data in a format that delivers user understandability, query performance, + resilience to change**

#### Star Schemas Versus OLAP Cubes

* **Dimensional models implemented in RDBMSs** = **star schemas**(resemblance to a star-like structure)
* Dimensional models implemented in *multidimensional* database environments are referred to as **online analytical processing (OLAP) cubes**
* If a DW/BI environment includes *either* star schemas or OLAP cubes, it leverages dimensional concepts
* Both **stars and cubes have a common logical design with recognizable dimensions**; however, the **physical implementation differs**



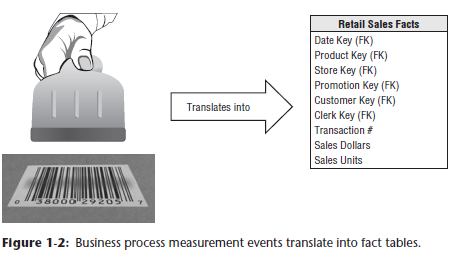
* When **data is loaded into an OLAP cube**, it is **stored and indexed using formats + techniques designed for dimensional data**
* **Performance aggregations or precalculated summary tables** are often **created and managed by the OLAP cube engine**
* Consequently, **cubes deliver superior query performance because of the pre-calculations, indexing strategies, + other optimizations**
* Business **users can drill down/up by adding/removing attributes from their analyses with excellent performance without issuing new** **queries**
* **OLAP cubes = provide more analytically robust functions that exceed those available w/ SQL**
* **Downside = pay a load performance price for these capabilities, especially w/ large data sets**
* Although the capabilities of OLAP technology are continuously improving, generally recommended that **detailed, atomic information be loaded into a star schema** while ***optional* OLAP cubes are then populated from the star schema**
* Most dimensional modeling techniques in this book = couched in terms of a relational star schema

##### OLAP Deployment Considerations

* Some things to keep in mind if you deploy data into OLAP cubes:
* **A star schema hosted in an RDB = a good physical foundation for building an OLAP cube, and is generally regarded as a more stable basis for backup + recovery**
* OLAP cubes have traditionally been noted for extreme performance advantages over RDBMSs, but **that distinction has become less important with advance**s **in CPU hardware**, such as appliances + in-memory databases, and RDBMS software, such as columnar databases
* OLAP cube data structures = more variable across different vendors than RDBMSs, thus the **final deployment details often depend on which OLAP vendor is chosen**
* It is typically more difficult to port BI applications between different OLAP tools than to port BI applications across different RDBs
* OLAP cubes typically offer more sophisticated security options than RDBMSs, such as limiting access to detailed data but providing more open access to summary data
* **OLAP cubes offer significantly richer analysis capabilities than RDBMSs, which are saddled by the constraints of SQL**
* This may be the main justification for using an OLAP product
* **OLAP cubes gracefully support slowly changing dimension type 2 changes** (Chapter 5: Procurement)
* *But* **cubes often need to be reprocessed partially or totally whenever data is overwritten using alternative slowly changing dimension techniques**
* OLAP **cubes gracefully support transaction and periodic snapshot fact** **tables**, but do **NOT handle accumulating snapshot fact tables because of the limitations on overwriting data as just described**
* OLAP cubes typically support complex ragged hierarchies of indeterminate depth, such as organization charts or bills of material, using native query syntax superior to the approaches required for RDBMSs
* OLAP cubes may impose detailed constraints on the structure of dimension keys that implement drill-down hierarchies compared to RDBs
* Some OLAP products do NOT enable dimensional roles or aliases, thus requiring separate physical dimensions to be defined

#### Fact Tables for Measurements

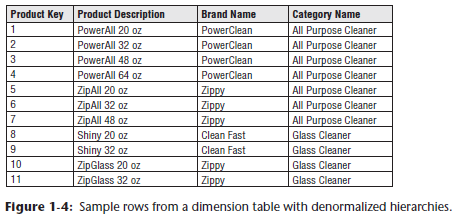
* **2 key components of a star schema = Fact Tables and Dimension tables**
* **Fact Table**in a dimensional model **stores the performance measurements resulting from an organization’s business process events**
* **Should strive to store the low-level measurement data resulting from a business process in a single dimensional model**
* Because **measurement data** is **overwhelmingly** the **largest set of data**, it **should NOT be replicated in multiple places for multiple organizational functions around an enterprise**
* Allowing **business users from multiple organizations** to **access a *single centralized repository*** for each set of measurement data **ensures the use of consistent data throughout an enterprise**
* ***“Fact”***represents a **business measure**
* Imagine standing in the marketplace watching products being sold and writing down the unit quantity and dollar sales amount for each product in each sales transaction
* These measurements are captured as products are scanned at the register



* **Each row in a fact table corresponds to a measurement event**
* The data on each row is at a specific level of detail, referred to as the **grain**, such as one row per product sold on a sales transaction
* **One of the core tenets of dimensional modeling** is that **all the measurement rows in a fact table must be at the same grain**
* Having the discipline to create **fact tables w/ a single level of detail ensures that measurements aren’t inappropriately double-counted**
* **NOTE:** **A measurement event in the physical world having a 1-to-1 relationship to a single row in a corresponding fact table = a bedrock principle for dimensional modeling (Everything else builds from this foundation)**
* The **most useful facts are numeric and additive**, such as dollar sales amount.
* In this book, use $ as the standard currency to make case study examples more tangible
* **Additivity** is **crucial because BI applications rarely retrieve a single fact table row**
* Rather, they **bring back hundreds, thousands, or even millions of fact rows at a time, and the most useful thing to do with so many rows is to add them up**
* No matter how the user slices data, sales units + dollars sum to a valid total
* We will see that **facts are sometimes semi-additive** (**cannot be summed across the *time* dimension**, such as account balances) or even **nonadditive** (**can *never* be added**, such as unit prices)
* You are forced to use counts and averages or are reduced to printing out the fact rows one at a time (an impractical exercise with a billion-row fact table)
* **Facts** are often described as **continuously valued** to help sort out what is a fact vs. a dimension attribute
* The dollar sales amount fact is continuously valued in this example because it can take on virtually any value within a broad range
* As an observer, you must stand out in the marketplace + wait for the measurement before you have any idea what the value will be
* **It’s theoretically possible for a measured fact to be textual; however, the condition rarely arises**
* In most cases, a textual measurement is a description of something + is drawn from a discrete list of values
* **A designer should make every effort to put textual data into dimensions where they can be correlated more effectively with the other textual dimension attributes and consume much less space**
* You should **NOT store redundant textual information in fact tables**
* **Unless the text is unique for *every* row in the fact table, it belongs in the dimension table**
* **A *true* text fact is rare because the unpredictable content of a text fact, like a freeform text comment, makes it nearly impossible to analyze**
* If there is no sales activity for a given product, you don’t put any rows in the table
* It is **important to not try to fill a fact table with 0’s representing no activity because these 0’s would overwhelm most fact tables**
* By **including only *true* activity, fact tables tend to be quite sparse**
* **Despite their sparsity, fact tables usually make up 90%+ of the total space consumed by a dimensional model**
* **Fact tables** tend to be **deep in terms of the number of rows**, **but narrow in terms of the number of columns**
* Given their size, **be judicious about fact table space utilization**
* **All fact table grains fall into 1 of 3 categories**: **transaction**, **periodic** **snapshot**, and **accumulating snapshot**
* Transaction grain fact tables = the most common, seen in Chapter 3: Retail Sales
* Both periodic and accumulating snapshots = Chapter 4: Inventory
* **All fact tables have 2 or more foreign keys** **(FK)** that **connect to the dimension tables’ primary keys (PK)**
* Ex: The product key in the fact table always matches a specific product key in the product dimension table
* **When all keys in a fact table correctly match their respective PK’s in the corresponding dimension tables, the tables satisfy** **referential integrity**
* You **access the fact table via the dimension tables joined to it**
* The **fact table generally has its own PK composed of a subset of the FK’s, often called a composite key**
* **Every table that has a composite key is a fact table**
* **Fact tables express many-to-many relationships + all others are dimension tables**
* There are **usually a handful of dimensions that together uniquely ID each fact table row**
* After this subset of the overall dimension list has been identified, **the rest of the dimensions take on a single value in the context of the fact table row’s PK**
* In other words, they go along for the ride

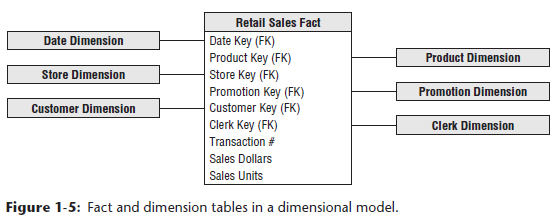
#### Dimension Tables for Descriptive Context

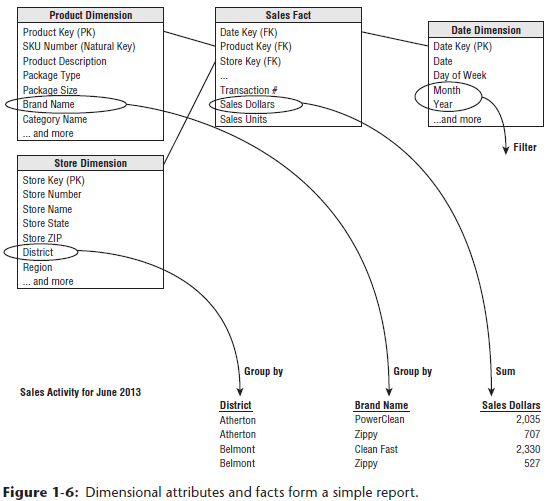
* **Dimension tables**are **integral companions to a fact table that** **contain the textual context (nouns) associated with a business process measurement event 🡪 describe the “who, what, where, when, how, and why” associated with an event**
* Dimension tables **often have many columns/attributes**
* Not uncommon for a dimension table to have 50 to 100 attributes; although, some dimension tables naturally have only a handful of attributes.
* **Dimension tables tend to have fewer rows than fact tables, but can be wide w/ many large text columns**
* **Each dimension is defined by a single PK**, which **serves as the basis for referential integrity** with any given fact table to which it is joined
* **Dimension attributes** serve as the **primary source of query constraints, groupings, + report labels**
* In a query or report request, **attributes are identified as the *BY* words**
* Ex: When a user wants to see dollar sales by brand, brand must be available as a dimension attribute
* Dimension table attributes play a vital role in the DW/BI system
* **Because they are the source of virtually all constraints and report labels, dimension attributes are critical to making the DW/BI system usable and understandable**
* **Attributes should consist of *real words* rather than cryptic abbreviations** 🡪 **strive to minimize the use of codes in dimension tables by replacing them w/ more verbose textual attributes**
* Should make standard decodes for operational codes available as dimension attributes to provide consistent labeling on queries, reports, + BI applications
* The decode values should never be buried in the reporting applications where inconsistency is inevitable
* Sometimes operational codes or identifiers have legitimate business significance to users or are required to communicate back to the operational world
* In these cases, the codes should appear as explicit dimension attributes, in addition to the corresponding user-friendly textual descriptors
* **Operational codes sometimes have intelligence embedded in them**
* Ex: The first 2 digits may identify the line of business, whereas the next 2 digits may identify the global region
* **Rather than forcing users to interrogate or filter on substrings within the operational codes, pull out the embedded meanings and present them to users as separate dimension attributes that can easily be filtered, grouped, or reported.**
* **In many ways, the DW is only as good as the dimension attributes, and the analytic power of the DW/BI environment is directly proportional to the quality + depth of the dimension attributes**
* The more time spent **providing attributes w/ verbose business terminology**, the better
* The more time spent **populating the domain values in an attribute column**, the better
* The more time spent **ensuring the quality of the values in an attribute column**, the better
* **Robust dimension attributes deliver robust analytic slicing-and-dicing capabilities**
* **NOTE**: **Dimensions provide the *entry points* to the data, and the final labels + groupings on all DW/BI analyses**
* When triaging operational source data, **it is sometimes unclear whether a numeric data element is a fact or dimension attribute**
* You often **make the decision by asking whether the column is a measurement that takes on lots of values + participates in calculations (making it a fact)**
* **Or is it a discretely valued description that is more or less constant + participates in constraints + row labels (making it a dimensional attribute)**
* Ex: The standard cost for a product seems like a constant attribute of the product but may be changed so often that you decide it is more like a measured fact
* **Occasionally, you can’t be certain of the classification; it is possible to model the data element either way (or both ways) as a matter of the designer’s prerogative**
* **NOTE**: The designer’s dilemma of whether a numeric quantity is a fact or a dimension attribute is rarely a difficult decision
* **Continuously valued numeric observations are almost always facts; discrete numeric observations drawn from a small list are almost always dimension attributes**
* **Dimension tables often represent hierarchical relationships**



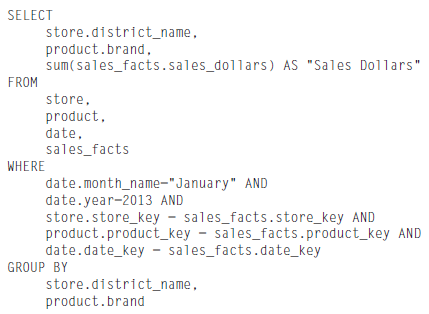
* Ex: Products roll up into brands + then into categories
* For each row in the product dimension, store the associated brand + category description
* **The hierarchical descriptive information is stored redundantly in the spirit of ease of use + query performance**
* **Resist the perhaps habitual urge to normalize data by storing only the brand code in the product dimension + creating a separate brand lookup table, + likewise for the category description in a separate category lookup table**
* Such normalization is called **snowflaking**
* Instead of 3NF, **dimension tables typically are highly denormalized with flattened many-to-one relationships within a single dimension table**
* **Because dimension tables typically are geometrically smaller than fact tables, improving storage efficiency by normalizing or snowflaking has virtually no impact on the overall database size**
* **You should almost always trade off dimension table space for simplicity + accessibility**

#### Facts and Dimensions Joined in a Star Schema

* Now that you understand fact + dimension tables, it’s time to bring the building blocks together in a dimensional model
* 
* **Each business process is represented by a dimensional model that consists of a fact table containing the event’s numeric measurements**
* **This fact table is surrounded by a halo of dimension tables that contain the textual context that was true at the moment the event occurred**
* This characteristic star-like structure is often called a **star join**, a term dating back to the earliest days of RDBs.
* The 1st thing to notice about the **dimensional schema** is its **simplicity + symmetry**
* Obviously, **business users benefit from the simplicity because the data is easier to understand + navigate**
* Charm of the above design in is that it is **highly recognizable to business users**
* The **reduced number of tables + use of meaningful business descriptors make it easy to navigate + less likely that mistakes will occur**
* The **simplicity of a dimensional model also has performance benefits**
* **Database optimizers process these simple schemas with fewer joins more efficiently**
* A **database engine can make strong assumptions about first constraining the heavily indexed dimension tables, and then attacking the fact table all at once with the Cartesian product of the dimension table keys satisfying the user’s constraints.**
* Amazingly, **using this approach, the optimizer can evaluate arbitrary n-way joins to a fact table in a single pass through the fact table’s index**
* Finally, **dimensional models are gracefully extensible to accommodate change**
* The predictable framework of a dimensional model withstands unexpected changes in user behavior
* **Every dimension is equivalent; all dimensions are symmetrically equal entry points into the fact table**
* The **dimensional model has no built-in bias regarding expected query patterns** + there are no preferences for the business questions asked this month vs. the questions asked next month
* You certainly don’t want to adjust schemas if business users suggest new ways to analyze their business
* **The most granular or atomic data has the most dimensionality**
* **Atomic data** **that has *not* been aggregated is the most expressive data** and **should be the foundation for every fact table design to withstand business users’ ad hoc attacks in which they pose unexpected queries**
* **With dimensional models, you can *add completely new dimensions* to the schema as long as a single value of that dimension is defined for each existing fact row**
* **Likewise**, you ***can add new facts to the fact table*, assuming that the level of detail is consistent with the existing fact table**
* You **can *supplement preexisting dimension tables* with new, unanticipated attributes**
* In each case above, **existing tables can be changed in place either by simply adding new data rows in the table or by executing an SQL ALTER TABLE command**
* Data would not need to be reloaded, + existing BI applications would continue to run without yielding different results
* We examine this graceful extensibility of dimensional models more fully in Chapter 3
* Another way to think about the complementary nature of fact + dimension tables is to see them translated into a report
* **Dimension attributes supply report filters + labeling, whereas fact tables supply numeric values**



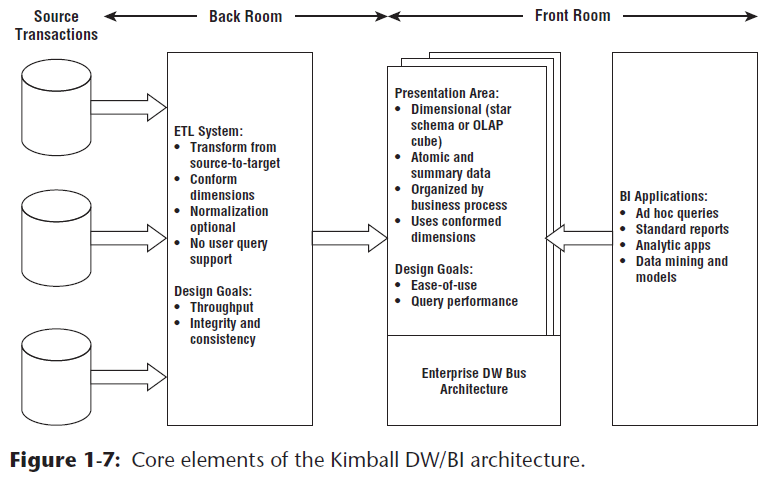
* You can easily envision the SQL written (or likely generated by a BI tool) to create this report:



* If you study this code snippet line-by-line, the first 2 lines under SELECT identify the dimension attributes in the report, followed by the aggregated metric from the fact table.
* The FROM clause identifies all the tables involved in the query
* The first 2 lines in the WHERE clause declare the report’s filter, + the remainder declare the JOINs between the dimension and fact tables
* Finally, the GROUP BY clause establishes the aggregation within the report.

### Kimball’s DW/BI Architecture

* Let’s investigate the components of a DW/BI environment based on the **Kimball architecture**
* Need to learn the strategic significance of each component to avoid confusing their role + function



* There are 4 separate + distinct components to consider in the DW/BI environment:
* **1) Operational source systems**
* **2) ETL system**
* **F**
* **F**

#### 1) Operational Source Systems

* Operational systems of record **capture the business’s transactions**.
* Think of the source systems as **outside the DW** because presumably you have **little/no control over the content and format of the data in these operational systems**
* The **main priorities** of the source systems are **processing performance and availability**
* **Operational queries** against source systems are **narrow, one-record-at-a-time queries** that are **part of the normal transaction flow** + are **severely restricted in their demands** on the operational system
* It is safe to assume that source systems are *not* queried in the broad + unexpected ways that DW/BI systems typically are queried
* Source systems **maintain little historical data**
* A **good DW can relieve the source systems of much of the responsibility for representing the past**
* In many cases, source systems are **special purpose applications without any commitment to sharing common data** such as product, customer, geography, or calendar **with other operational systems in the organization**
* Of course, a broadly adopted cross-application **enterprise resource planning (ERP) system** or **operational master data management system** could help address these shortcomings

#### 2) Extract, Transformation, and Load (ETL) System

* Consists of a **work area**, **instantiated data structures**, and a **set of processes**
* The ETL system is ***everything* between the operational source systems and the DW/BI presentation area** (elaborated on in Chapter 19: ETL Subsystems and Techniques)
* **Extraction** = 1st step in the process of getting data into the DW environment
* This means reading and understanding the source data + copying the data needed into the ETL system for further manipulation
* At this point, the data belongs to the data warehouse.
* After data is extracted to the ETL system, there are numerous potential **transformations**, such as cleansing the data (correcting misspellings, resolving domain conflicts, dealing w/ missing elements, or parsing into standard formats), combining data from multiple sources, and de-duplicating data
* The **ETL system adds value to the data with these cleansing and conforming tasks by changing the data and enhancing it**
* In addition, these **activities can be architected to create** **diagnostic metadata**, eventually **leading to business process reengineering to improve data quality in *source* systems over time**
* Final step of the ETL process = the **physical structuring + loading** of data **into the presentation area’s target dimensional models**
* Because the primary mission of the ETL system is to hand off the dimension + fact tables in the delivery step, these subsystems are *critical*
* **Many of these defined subsystems focus on dimension table processing**, such as surrogate key assignments, code lookups to provide appropriate descriptions, splitting, or combining columns to present the appropriate data values, or joining underlying 3NF table structures into flattened denormalized dimensions
* In **contrast, fact tables are typically large + time consuming to load, but preparing them for the presentation area is typically straightforward**
* When the dimension + fact tables in a dimensional model have been updated, indexed, supplied with appropriate aggregates, + further quality assured, the business community is notified that the new data has been published
* There remains industry consternation about whether the data in the ETL system should be repurposed into physical normalized structures prior to loading into the presentation area’s dimensional structures for querying and reporting
* ETL system is typically dominated by the simple activities of sorting + sequential processing
* In many cases, the ETL system is *not* based on relational technology but instead may rely on a system of flat files
* After validating the data for conformance with the defined one-to-one and many-to-one business rules, it may be pointless to take the final step of building a 3NF physical database, just before transforming the data once again into denormalized structures for the BI presentation area
* However, there *are* cases in which the data arrives at the doorstep of the ETL system in a 3NF relational format
* In these situations, the ETL system developers may be more comfortable performing the cleansing and transformation tasks using normalized structures
* Although a normalized database for ETL processing is acceptable, there are some reservations about this approach
* **The creation of *both* normalized structures for the ETL *and* dimensional structures for presentation means that the data is potentially extracted, transformed, and loaded *twice***(once into the normalized database and then *again* when you load the dimensional model)
* Obviously, this two-step process requires more time and investment for the development, more time for the periodic loading or updating of data, + more capacity to store the multiple copies of the data
* At the bottom line, this typically translates into the need for larger development, ongoing support, and hardware platform budgets.
* Unfortunately, **DW/BI initiatives can fail miserably b/c orgs focused all energy + resources on constructing the normalized structures rather than allocating time to developing a *dimensional presentation area* that *supports improved business decision making***
* Although **enterprise-wide data consistency is a fundamental goal** of the DW/BI environment, there may be effective + less costly approaches than physically creating normalized tables in the ETL system, if these structures don’t already exist
* **NOTE**: It is *acceptable* to create a normalized database to support the ETL processes; however, *this is not the end goal*
* The normalized structures must be off-limits to user queries because they defeat the **twin goals of understandability and performance**

#### 3) Data Presentation Area (to Support BI)

* Where **data is organized, stored, + made available for direct querying by users, report writers, + other analytical BI applications.**
* Because the back room ETL system is off-limits, the **presentation area is the DW/BI environment as far as the business community is concerned**; it is all the business sees + touches via their access tools + BI applications
* The **presentation area with its dimensional models is all about getting the data *out***
* Strong opinions about the presentation area:
* 1) Insist that **data be presented, stored, and accessed in dimensional schemas, either relational star schemas or OLAP cubes**
* Fortunately, the industry has matured + has concluded that dimensional modeling is the most viable technique for delivering data to DW/BI users
* 2) It must **contain** **detailed, atomic data**
* **Atomic data (lowest level of detail)** is required to withstand assaults from unpredictable ad hoc user queries
* Although the **presentation area also may contain performance-enhancing aggregated data, it is not sufficient to deliver these summaries without the underlying granular data in a dimensional form**
* Completely **unacceptable to store only summary data in dimensional models while the atomic data is locked up in normalized models**
* Impractical to expect a user to drill down through dimensional data almost to the most granular level + then lose the benefits of a dimensional presentation at the final step. Although DW/
* BI users + applications may look infrequently at a single line item on an order, they may be very interested in last week’s orders for products of a given size (or flavor, package type, or manufacturer) for customers who first purchased w/in the last 6 months (or reside in a given state or have certain credit terms)
* **The most finely grained data *must* be available in the presentation area so that users can ask the most precise questions possible**
* Because **users’ requirements are unpredictable + constantly changing**, you **must provide access to the exquisite details so they can roll up to address the questions of the moment**
* The **presentation data area should be structured around *business process measurement events***
* This approach naturally aligns with the operational source data capture systems.
* **Dimensional models should correspond to physical data capture events + should *not* be designed to deliver the report-of-the-day**
* An enterprise’s business processes cross the boundaries of organizational departments + functions.
* In other words, you should **construct a *single* fact table for atomic sales metrics rather than populating separate similar, but slightly different, databases containing sales metrics for the sales, marketing, logistics, and finance teams**
* All **dimensional structures must be built using common, conformed dimensions**
* This is the basis of the **enterprise data warehouse bus architecture**(Chapter 4)
* 3) **Adherence to the bus architecture** = the final stake in the ground for a presentation area
* **Without shared, conformed dimensions, a dimensional model becomes a standalone application**
* **Isolated stovepipe data sets** that cannot be tied together are the bane of the DW/BI movement as they **perpetuate incompatible views of the enterprise**
* **If you have any hope of building a robust + integrated DW/BI environment, you *must* commit to the enterprise bus architecture**
* **Dimensional models** **designed** **w/ conformed dimensions can be readily combined + used together**
* Presentation area in a large enterprise DW/BI solution ultimately consists of dozens of dimensional models w/ many associated dimension tables shared across fact tables
* **Using the bus architecture is the secret to building distributed DW/BI systems**
* When the **bus architecture** is used as a **framework**, you can **develop the enterprise data warehouse (EDW) in an agile, decentralized, realistically scoped, iterative manner**
* **NOTE**: **Data in** the query-able **presentation area** of the DW/BI system must be **dimensional**, **atomic** (complemented by performance-enhancing aggregates), **business process-centric**, and **adhere to the EDW bus architecture**, + **data must NOT be structured according to individual departments’ interpretation of the data**

#### 4) BI Applications

* Loosely refers to the **range of capabilities provided to business users to leverage the presentation area for analytic decision making**
* By definition, all BI applications query the data in the DW/BI presentation area
* **Querying**, obviously = the **whole point of using data for improved decision making**
* A BI application **can be as simple as an ad hoc query tool** or as **complex as a sophisticated data mining or modeling application**
* **Ad hoc query tools**, as **powerful** as they are, can be **understood + used effectively** **by only a small % of potential DW/BI business users**
* ***Most business users will likely access the data via prebuilt parameter-driven applications + templates that do not require users to construct queries directly***
* Some of the more sophisticated applications, such as modeling or forecasting tools, may upload results back into the operational source systems, ETL system, or presentation area

#### Restaurant Metaphor for the Kimball Architecture

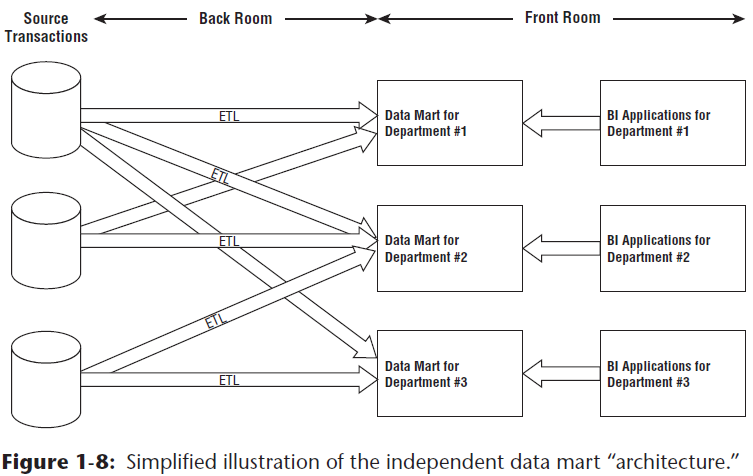
* **ETL in the Back Room Kitchen**
* ETL system = analogous to the kitchen of a restaurant
* A restaurant’s kitchen = a world unto itself where chefs take raw materials + transform them into appetizing, delicious meals for diners
* But long before a commercial kitchen swings into operation, **a significant amount of planning goes into designing the workspace layout and components**
* A kitchen is organized with several design goals in mind
* 1) Layout must be **highly efficient**
* Restaurant managers want high kitchen throughput
* When a restaurant is packed + everyone is hungry, there is no time for wasted movement
* 2) Delivering **consistent quality** from a kitchen = 2nd important goal
* An establishment = doomed if plates coming out of the kitchen repeatedly fail to meet expectations.
* To achieve consistency, chefs create special sauces once in the kitchen, rather than sending ingredients out to the table where variations will inevitably occur
* 3) Finally, a kitchen’s output (meals) must also be of **high integrity**
* Wouldn’t want someone to get food poisoning from dining at your restaurant.
* Consequently, kitchens = designed with integrity in mind (salad prep doesn’t happen on the same surfaces where raw chicken is handled)
* Just as **quality, consistency, + integrity** are major considerations when designing a restaurant kitchen, they are **also ongoing concerns for everyday management** of the restaurant
* Chefs strive to obtain the best raw materials possible
* Procured products must meet quality standards + are rejected if they don’t meet minimum standards
* Most fine restaurants modify their menus based on the availability of quality ingredients
* The staff = skilled professionals wielding the tools of their trade 🡪 cooks manipulate razor-sharp knives with incredible confidence + ease and also operate powerful equipment + work around extremely hot surfaces without incident
* Given the dangerous surroundings, the **back-room kitchen is off limits to patrons**
* Things happen in the kitchen that customers just shouldn’t see, it simply isn’t safe
* Professional cooks handling sharp knives shouldn’t be distracted by diners’ inquiries
* Also don’t want patrons entering a kitchen to dip fingers into a sauce to see whether they want to order an entrée
* To prevent these intrusions, most restaurants have a closed door that separates the kitchen from the area where diners are served
* Even restaurants that boast an open kitchen format typically have a barrier, such as a partial wall of glass, separating the 2 environments, where diners are invited to watch but can’t wander into the kitchen
* Although part of the kitchen may be visible, there are always out-of-view back rooms where the less visually desirable preparation occurs
* **A DW’s ETL system resembles the restaurant’s kitchen**
* **Source data** = **magically transformed** into **meaningful, presentable information**
* The back room **ETL system must be laid out + architected long before any data is extracted from the source**
* Like the kitchen, the **ETL system is designed to ensure throughput 🡪** must **transform raw source data into the target model efficiently, minimizing unnecessary movement**
* Obviously, an **ETL system = also highly concerned w/ data quality, integrity, + consistency**
* **Incoming data is checked** for reasonable **quality** as it enters
* **Conditions** are **continually monitored** to ensure ETL outputs are of high integrity.
* **Business rules to consistently derive value-add metrics + attributes are applied *once* by skilled professionals in the ETL system** rather than relying on each patron to develop them independently.
* Yes, this puts extra burden on the ETL team, but it’s done to deliver a better, more consistent product to the DW/BI patrons
* **NOTE**: A *properly designed* DW/BI environment trades off work in the front room BI applications in favor of work in the back room ETL system
* **Front room work must be done over + over by business users, whereas back-room work is done once by the ETL staff**
* Finally, **ETL system should be off limits to the business users and BI application developers**
* Don’t want busy ETL professionals distracted by unpredictable inquiries from BI users
* The consequences might be highly unpleasant if users dip proverbial fingers into interim staging pots while data prep is still in process
* Also, activities occur in the ETL system that the DW/BI patrons shouldn’t see
* When data is ready + quality checked for user consumption, it’s brought through the doorway into the DW/BI presentation area
* **Data Presentation and BI in the Front Dining Room**
* What are the key factors that differentiate restaurants?
* According to the popular restaurant ratings + reviews, restaurants are typically scored on 4 distinct qualities: Food (quality, taste, and presentation), Decor (appealing, comfortable surroundings for the patrons), Service (prompt food delivery, attentive support staff , and food received as ordered), + Cost
* Most patrons focus initially on the food score when they’re evaluating dining options
* First and foremost, does the restaurant serve good food? That’s the restaurant’s **primary deliverable**
* However, decor, service, + cost factors also affect patrons’ overall dining experience + are considerations when evaluating whether to eat at a restaurant.
* **Primary deliverable from the DW/BI kitchen = the data in the presentation area**
* What data is available? 🡪 Like a restaurant, a DW/BI system provides “menus” to describe what’s available via metadata, published reports, + parameterized analytic applications
* **DW/BI patrons expect consistency + high quality**
* **Presentation area’s data must be properly prepared + safe to consume**.
* Presentation area’s **decor should be organized for the patrons’ comfort**
* Must be **designed based on the preferences of the BI diners, *not* the development staff**
* **Service** is also critical in the DW/BI system
* **Data must be delivered, as ordered, promptly in a form that is appealing to the business user or BI application developer**
* Finally, **cost** is a factor for the DW/BI system
* Kitchen staff may be dreaming up elaborate, expensive meals, but if there’s no market at that price point, a restaurant won’t survive.
* If patrons like their dining experience, then everything is rosy for the manager
* The dining room is always busy; sometimes there’s even a waiting list
* The restaurant manager’s performance metrics are all promising: high numbers of diners, table turnovers, and nightly revenue and profit, while staff turnover is low
* Things look so good that the owner is considering an expansion site to handle the traffic
* On the other hand, if the restaurant’s diners aren’t happy, things go downhill in a hurry
* With a limited number of patrons, the restaurant isn’t making enough money to cover expenses, + staff isn’t making any tips
* In a relatively short time, the restaurant closes.
* Restaurant managers often **proactively check on their diners’ satisfaction** w/ the food + dining experience
* If a patron is unhappy, they take immediate action to rectify the situation
* Similarly, **DW/BI managers should proactively monitor satisfaction**.
* Can’t afford to wait to hear complaints
* Often, people will abandon a restaurant without even voicing concerns
* Over time, managers notice that diner counts have dropped but may not even know why
* **Inevitably, the prior DW/BI patrons will locate another “restaurant” that better suits their needs + preferences, wasting the millions of dollars invested to design, build, + staff the DW/BI system**
* Of course, you can **prevent this unhappy ending by managing proactively**; make sure the kitchen is properly organized + utilized to deliver as needed to the presentation area’s food, decor, service, + cost

### Alternative DW/BI Architectures

* There are 2 two dominant alternatives to the Kimball architecture + there’s a hybrid approach that combines alternatives
* Fortunately, over the past few decades, the differences between the Kimball architecture + the alternatives have softened
* Even more fortunate, **there’s a role for dimensional modeling regardless of your architectural predisposition**

#### Independent Data Mart Architecture

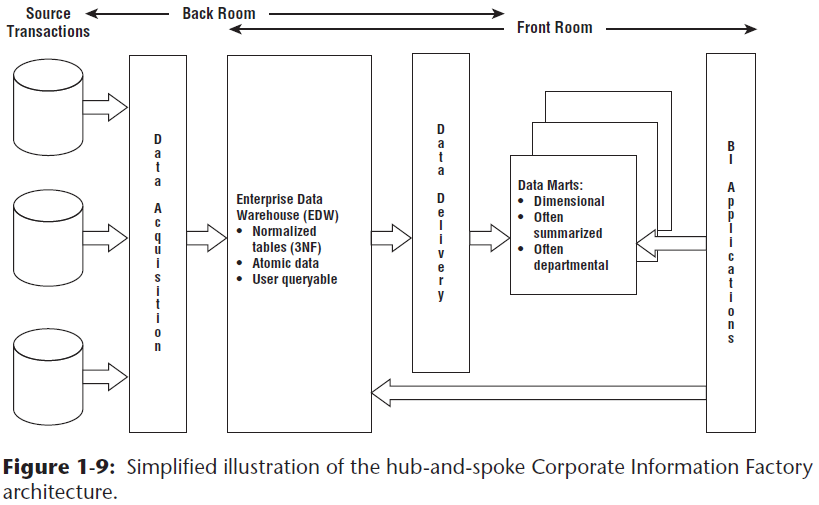
* With this approach, *analytic data is deployed on a departmental basis without concern to sharing + integrating information across the enterprise*, as illustrated below



* Typically, a single department identifies requirements for data from an operational source system
* The department works w/ IT staff or outside consultants to construct a database that satisfies *their* departmental needs, reflecting *their* business rules + preferred labeling
* Working in **isolation**, this departmental data mart addresses the department’s analytic requirements
* Meanwhile, *another* department is interested in the same source data
* Extremely common for multiple departments to be interested in the same performance metrics resulting from an organization’s core business process events
* But because this department doesn’t have access to the data mart initially constructed by the other department, it proceeds down a similar path on its own, obtaining resources + building a departmental solution that contains similar, but slightly different data
* *When business users from these 2 departments discuss organizational performance based on reports from their respective repositories, not surprisingly, none of the numbers match because of the differences in business rules and labeling*
* These standalone analytic silos represent a DW/BI “architecture” that’s essentially un-architected
* **Although no industry leaders advocate these independent data marts, this approach is prevalent, especially in large organizations**
* It mirrors the way many organizations fund IT projects, plus it requires zero cross-organizational data governance + coordination
* It’s the path of least resistance for fast development at relatively low cost, *at least in the short run*
* Of course, multiple uncoordinated extracts from the same operational sources + redundant storage of analytic data are *inefficient and wasteful in the long run*
* *Without any enterprise perspective, this independent approach results in myriad standalone point solutions that perpetuate incompatible views of the organization’s performance, resulting in unnecessary organizational debate and reconciliation*
* Often independent data marts have embraced dimensional modeling because they’re interested in delivering data that’s easy for the business to understand + highly responsive to queries
* So, our **concepts of dimensional modeling are often applied in this architecture, despite the complete disregard for some of our core tenets, such as focusing on atomic details, building by business process instead of department, + leveraging conformed dimensions for enterprise consistency and integration**

#### Hub-and-Spoke Corporate Information Factory Inmon Architecture

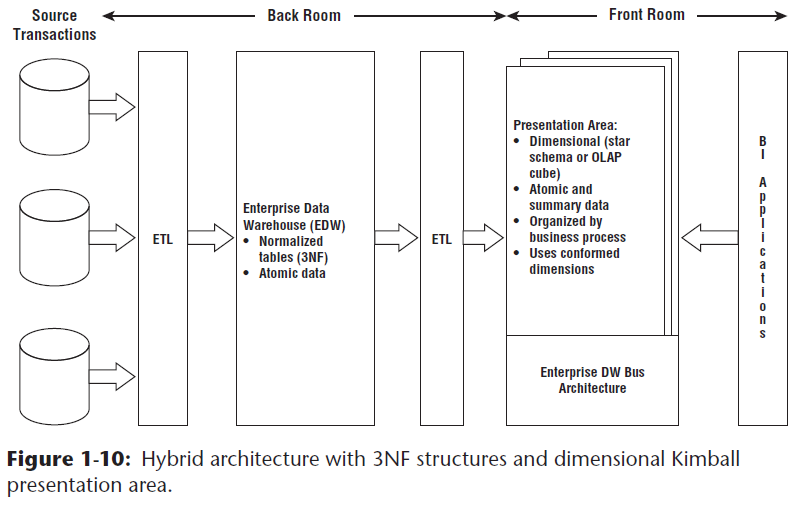
* The **hub-and-spoke Corporate Information Factory (CIF) approach** is advocated by Bill Inmon + others in the industry
* See a simplified version of the CIF, focusing on the core elements + concepts that warrant discussion



* With the CIF, data is extracted from the operational source systems + processed through an ETL system sometimes referred to as **data acquisition**
* *The atomic data that results from this processing lands in a 3NF database* 🡪 a normalized, atomic repository referred to as the EDW within the CIF architecture
* **Although the Kimball architecture enables *optional* normalization to support ETL processing, the normalized EDW is a mandatory construct in the CIF**
* **Like the Kimball approach, the CIF advocates enterprise data coordination + integration**
* The CIF says the normalized EDW fills this role, whereas the **Kimball architecture stresses the importance of an enterprise bus with conformed dimensions**
* **NOTE:** **The process of normalization does not technically speak to integration.**
* **Normalization** = just **creates physical tables that implement many-to-one relationships**.
* **Integration** = **requires that inconsistencies arising from separate sources be resolved**
* *Separate incompatible database sources can be normalized to the hilt without addressing integration*
* **Kimball architecture based on conformed dimensions reverses this logic + focuses on resolving data inconsistencies without explicitly requiring normalization**
* Organizations who have adopted the CIF approach often have business users accessing the EDW repository due to its level of detail or data availability timeliness
* However, subsequent ETL data delivery processes also populate downstream reporting + analytic environments to support business users
* Although often dimensionally structured, the resultant analytic databases typically differ from structures in the Kimball architecture’s presentation area in that they’re frequently *departmentally*-centric (rather than organized around business processes) + populated with *aggregated* data (rather than atomic details)
* If the data delivery ETL processes apply business rules beyond basic summarization, such as departmental renaming of columns or alternative calculations, it may be difficult to tie these analytic databases to the EDW’s atomic repository
* **NOTE**: The most extreme form of a *pure* CIF architecture is unworkable as a DW, in Kimball’s opinion
* *Such an architecture locks the atomic data in difficult-to-query normalized structures, while delivering departmentally incompatible data marts to different groups of business users*

#### Hybrid Hub-and-Spoke and Kimball Architecture

* The marriage of the Kimball + Inmon CIF architectures populates **a CIF-centric EDW that is completely off-limits to business users for analysis + reporting, + it’s merely the source to populate a Kimball-esque presentation area in which the data is dimensional, atomic (complemented by aggregates), process-centric, + conforms to the EDW bus architecture**
* Some proponents of this blended approach claim it’s the best of both worlds
* Yes, it blends the two enterprise-oriented approaches + may leverage a preexisting investment in an integrated repository, while addressing the performance + usability issues associated with the 3NF EDW by offloading queries to the dimensional presentation area
* And because the end deliverable to the business users and BI applications is constructed based on Kimball tenets, who can argue with the approach?



* If already invested in the creation of a 3NF EDW, but it’s not delivering on the users’ expectations of fast and flexible reporting and analysis, this hybrid approach might be appropriate for your org.
* If starting with a blank sheet of paper, the **hybrid approach will likely cost more time + money, both during development + ongoing operation, given the multiple movements of data + redundant storage of atomic details**
* If you have the appetite, perceived need, + perhaps most important, budget + organizational patience to fully normalize + instantiate your data before loading it into dimensional structures that are well designed according to the Kimball methods, go for it

### Dimensional Modeling Myths

* Despite widespread acceptance of dimensional modeling, some misperceptions persist in industry
* These false assertions are a distraction, especially when you want to align your team around common best practices
* **Myth 1: Dimensional Models are Only for Summary Data**
* This is frequently the root cause of ill-designed dimensional models.
* Because you **can’t possibly predict all questions asked by business users, you need to provide them with query-able access to the most detailed data** so they can roll it up based on the business question
* **Data at the lowest level of detail is practically impervious to surprises or changes**
* **Summary data** should ***complement*** the **granular detail** solely to provide **improved performance** for common queries, but ***not* replace the details**
* **NOTE:** A related corollary to this 1st myth = only a limited amount of historical data should be stored in dimensional structures
* *Nothing about a dimensional model prohibits storing substantial history*
* The **amount of history available in dimensional models must only be driven by the business’s requirements**
* **Myth 2: Dimensional Models are Departmental, Not Enterprise**
* Rather than drawing boundaries based on organizational departments, **dimensional models should be organized around business processes**, such as orders, invoices, + service calls.
* **Multiple business functions often want to analyze the same metrics resulting from a single business process**
* **Multiple extracts of the same source data that create multiple, inconsistent analytic databases should be avoided**
* **Myth 3: Dimensional Models are Not Scalable**
* **Dimensional models = *extremely* scalable**
* Fact tables frequently have billions of rows (ones containing 2 trillion rows have been reported)
* Database vendors have wholeheartedly embraced DW/BI + continue to incorporate capabilities into their products to optimize dimensional models’ scalability + performance
* Both normalized + dimensional models contain the same information + data relationship (the logical content is identical)
* Every data relationship expressed in one model can be accurately expressed in the other
* Both normalized and dimensional models can answer exactly the same questions, albeit with varying difficulty
* **Myth 4: Dimensional Models are Only for Predictable Usage**
* **Dimensional models should NOT be designed by focusing on pre-defined reports or analyses**
* The **design should center on measurement processes**
* Obviously, it’s important to consider the BI application’s filtering + labeling requirements
* But you shouldn’t design for a top ten list of reports in a vacuum because **the list is bound to change**, making the dimensional model a moving target
* **Key = focus on the organization’s measurement events that are typically stable, *unlike* analyses that are constantly evolving**
* A related corollary = Dimensional models aren’t responsive to changing business needs
* On the contrary, because of their symmetry, dimensional structures are extremely flexible + adaptive to change
* **Secret to query flexibility = building fact tables at the most granular level**
* Dimensional models that deliver only summary data are bound to be problematic, as users run into analytic brick walls when they try to drill down into details not available in the summary tables
* Developers also run into brick walls because they can’t easily accommodate new dimensions, attributes, or facts with these prematurely summarized tables
* **Correct starting point for dimensional models = to express data at the lowest detail possible for maximum flexibility and extensibility**
* Remember, **when you pre-suppose the business question, you’ll likely pre-summarize the data, which can be fatal in the long run**
* **Delivering dimensional models populated with the most detailed data possible ensures maximum flexibility + extensibility**
* **Delivering anything less** **undermines the foundation necessary for robust BI**
* **Myth 5: Dimensional Models Can’t Be Integrated**
* Dimensional models most certainly **can be integrated *IF* they conform to the EDW bus architecture**
* **Conformed dimensions** are built + maintained as **centralized, persistent master data in the ETL system + then reused across dimensional models to enable data integration + ensure semantic consistency**
* **Data integration depends on standardized labels, values, and definitions**
* It is **hard work to reach organizational consensus + then implement the corresponding ETL rules**, but you **can’t dodge the effort**, regardless of whether you’re populating normalized or dimensional models.
* Presentation area databases that don’t adhere to the bus architecture w/ shared conformed dimensions lead to standalone solutions
* You can’t hold dimensional modeling responsible for organizations’ failure to embrace one of its fundamental tenets

### More Reasons to Think Dimensionally

* Most of this book = focus on dimensional modeling for designing databases in the DW/BI *presentation area*
* But **dimensional modeling concepts go beyond the design of simple and fast data structure**
* One should **think dimensionally at *other* critical junctures of a DW/BI project**.
* **When gathering requirements** for a DW/BI initiative, you need to **listen for + then synthesize the findings around *business processes***
* Sometimes teams get lulled into focusing on a set of required reports or dashboard gauges.
* Instead, **constantly ask yourself about the business process measurement events *producing* the report or dashboard metrics**
* When specifying a project’s scope, you stand firm to focus on a *single* business process per project and do NOT sign up to deploy a dashboard that covers a handful of them in a single iteration
* Critical that the DW/BI team **concentrates on *business processes*** + it’s **equally important to get IT + business management on the same wavelength**
* Due to historical IT funding policies, the business may be more familiar with departmental data deployments
* Shift their mindset about the DW/BI rollout to a *process* perspective
* **When prioritizing opportunities and developing the DW/BI roadmap, business processes are the unit of work**
* Fortunately, business management typically embraces this approach because it mirrors their thinking about KPI’s
* Plus, they’ve lived with the inconsistencies, incessant debates, + never ending reconciliations caused by the departmental approach, so they’re ready for a fresh tactic
* **Working w/ business leadership partners, rank each business process on business value + feasibility, then tackle processes with the highest impact + feasibility scores first**
* **Although prioritization is a joint activity with the business, underlying understanding of the organization’s business processes is essential to its effectiveness and subsequent actionability**
* **If tasked w/ drafting the DW/BI system’s data architecture, you need to wrap your head around the organization’s processes, *along* *with* the associated master descriptive dimension data**
* The prime deliverable for this activity, the EDW bus matrix, is vetted in Chapter 4
* The matrix also serves as a useful tool for touting the potential benefits of a more rigorous master data management (MDM) platform.
* **Data stewardship or governance programs should focus first on the major dimensions.**
* Depending on the industry, the list might include date, customer, product, employee, facility, provider, student, faculty, account, and so on
* **Thinking about the central nouns used to describe the business translates into a list of data governance efforts to be led by SMEs from the business community**
* Establishing data governance responsibilities for these nouns is the key to eventually deploying dimensions that deliver consistency + address the business’s needs for analytic filtering, grouping, + labeling
* **Robust dimensions translate into robust DW/BI systems**
* So, **the fundamental motivation for dimensional modeling is front + center long before you design star schemas or OLAP cubes**
* Likewise, the dimensional model will *remain* in the forefront during the subsequent ETL system + BI application designs
* **Dimensional modeling concepts link the business + technical communities together as they jointly design the DW/BI deliverables**

### Agile Considerations

* Currently (2013), there’s significant interest within the DW/BI industry on **agile development**practices, which **focus on manageably-sized increments of work that can be completed within reasonable timeframes measured in weeks, rather than tackling a much larger scoped (and hence riskier) project with deliverables promised in months or years**
* Many of the core tenets of agile methodologies align with Kimball best practices, including:
* **Focus on delivering business value** (the Kimball mantra for decades)
* Value **collaboration between the development team and business stakeholders**.
* Stress **ongoing face-to-face communication, feedback, + prioritization w/ the business stakeholders**
* **Adapt quickly** to inevitably **evolving requirements**
* Tackle development in an **iterative, incremental** manner
* Although this list is compelling, a **common criticism of the agile approaches is the lack of planning and architecture, coupled with ongoing governance challenges**
* The EDW bus matrix is a powerful tool to address these shortcomings.
* **The bus matrix provides a framework and master plan for agile development**, + identifies the reusable common descriptive dimensions that provide both data consistency + reduced time-to-market delivery
* With the right collaborative mix of business + IT stakeholders in a room, the EDW bus matrix can be produced in relatively short order
* Incremental development work can produce components of the framework until sufficient functionality is available + then released to the business community
* Some clients + students lament that although they want to deliver consistently defined conformed dimensions in their DW/BI environments, it’s “just not feasible.”
* They explain that they would if they could, but with the focus on agile development techniques, it’s “impossible” to take the time to get organizational agreement on conformed dimensions.
* We argue that **conformed dimensions enable agile DW/BI development, along with agile decision making**
* As you flesh out the portfolio of master conformed dimensions, the development crank starts turning faster and faster.
* The time-to-market for a new business process data source shrinks as developers reuse existing conformed dimensions
* Ultimately, new ETL development focuses almost exclusively on delivering more fact tables because the associated dimension tables are already sitting on the shelf ready to go
* Without a framework like the EDW bus matrix, some DW/BI teams have fallen into the trap of using agile techniques to create analytic or reporting solutions in a vacuum
* In most situations, the team worked with a small set of users to extract a limited set of source data + make it available to solve their unique problems
* The outcome is often a standalone data stovepipe that others can’t leverage, or worse yet, delivers data that doesn’t tie to the organization’s other analytic information
* **We encourage agility, when appropriate, however building isolated data sets should be avoided**
* As with most things in life, moderation and balance between extremes is almost always prudent