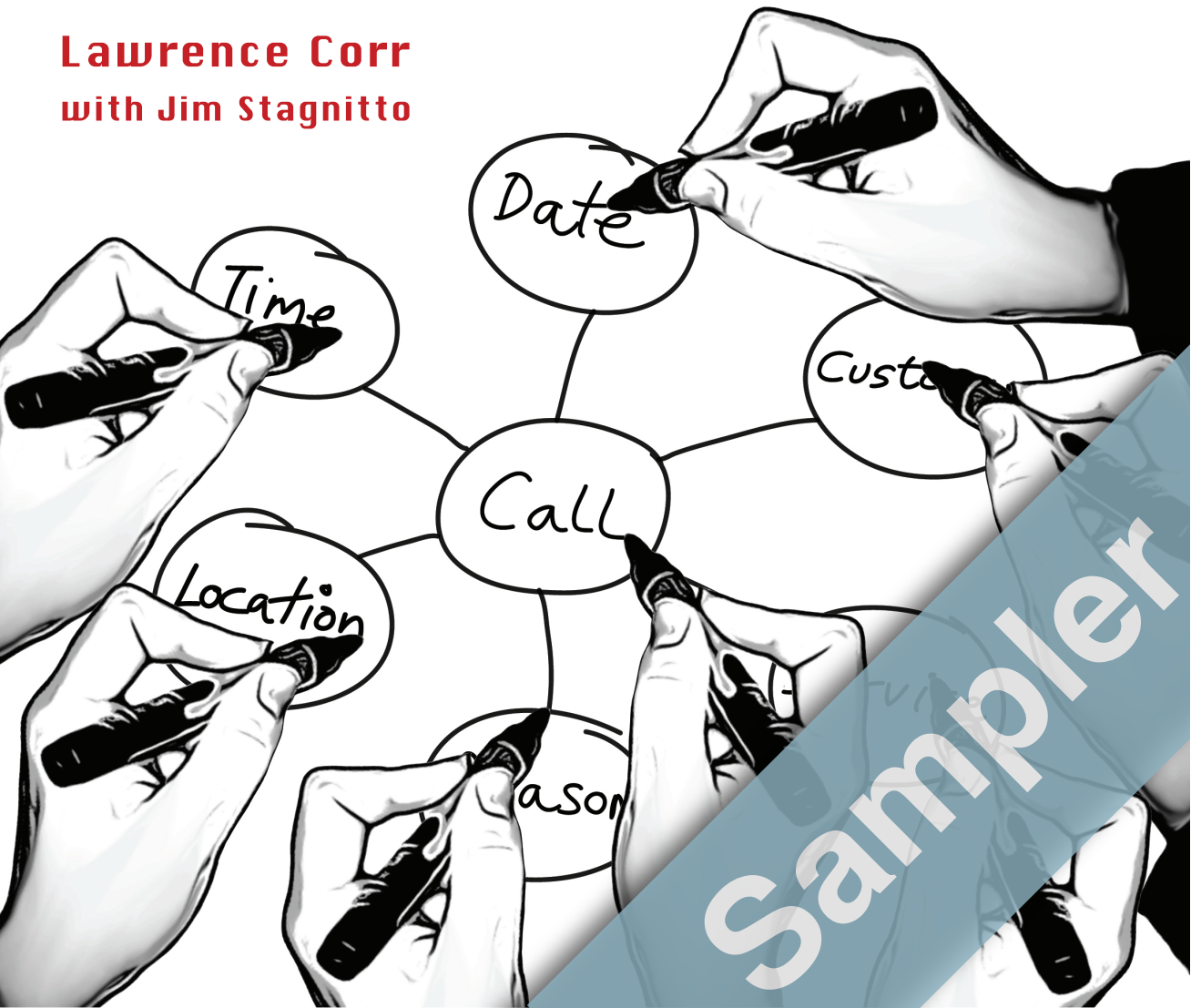
# Agile Data Warehouse Design

## Collaborative Dimensional Modeling, from Whiteboard to Star Schema

Lawrence Corr
with Jim Stagnitto



**Agile Data Warehouse Design** is a step-by-step guide for capturing data warehousing/business intelligence (DW/BI) requirements and turning them into high performance dimensional models in the most direct way: by *modelstorming* (data modeling + brainstorming) with BI stakeholders.

This book describes BEAM✲, an agile approach to dimensional modeling, for improving communication between data warehouse designers, BI stakeholders and the whole DW/BI development team. BEAM✲ provides tools and techniques that will encourage DW/BI designers and developers to move away from their keyboards and entity relationship based tools and model interactively with their colleagues. The result is everyone thinks dimensionally from the outset! Developers understand how to efficiently implement dimensional modeling solutions. Business stakeholders feel ownership of the data warehouse they have created, and can already imagine how they will use it to answer their business questions.

Within this book, you will learn:

- **Agile dimensional modeling using Business Event Analysis & Modeling (BEAM✲)**
- **Modelstorming: data modeling that is quicker, more inclusive, more productive, and frankly more fun!**
- **Telling dimensional data stories using the *7Ws* (*who, what, when, where, how many, why* and *how*)**
- ***Modeling by example* not abstraction; using data story themes, not crow's feet, to describe detail**
- **Storyboarding the data warehouse to discover conformed dimensions and plan iterative development**
- **Visual modeling: sketching timelines, charts and grids to model complex process measurement – simply**
- **Agile design documentation: enhancing star schemas with BEAM✲ dimensional shorthand notation**
- **Solving difficult DW/BI performance and usability problems with proven dimensional design patterns**

**Lawrence Corr** is a data warehouse designer and educator. As Principal of DecisionOne Consulting, he helps organizations to review and simplify their data warehouse designs, and advises on visual data modeling techniques. He regularly holds agile modeling workshops worldwide and has taught dimensional DW/BI skills to thousands of business/IT professionals.

**Jim Stagnitto** is a data warehouse and master data management architect specializing in the healthcare, financial services, and information service industries. He is the founder of the data warehousing and data mining consulting firm Llumino.

Decision Press
decisionone.co.uk

BEAM✲
modelstorming.com

ISBN 978-0-9568172-0-4
90000
9 780956 817204

# Agile Data Warehouse Design

## Collaborative Dimensional Modeling, from Whiteboard to Star Schema

Lawrence Corr
with Jim Stagnitto

Decision Press
decisionone.co.uk

**Agile Data Warehouse Design**

by Lawrence Corr with Jim Stagnitto






Published by DecisionOne Press, Burwood House, Leeds LS28 7UJ, UK.
Email: information@decisionone.co.uk, Tel: +44 7971 964824.

**Proofing:** Laurence Hesketh, Geoff Hurrell

**Illustrators:** Gill Guile and Lawrence Corr

**Cover Design:** After Escher

**Printing History:**

November 2011: First Edition. January 2012: Revision. October 2012: Revision. May 2013: Revision.

**Non-Printing History:**

May 2013: eBook First Edition.

Displayed on recycled pixels



ISBN: 978-0-9568172-0-4 [2013-05-10]

# About the Authors

**Lawrence Corr** is a data warehouse designer and educator. As Principal of DecisionOne Consulting, he helps organizations to improve their Business Intelligence systems through the use of visual data modeling techniques. He regularly teaches agile dimensional modeling courses worldwide and has taught DW/BI skills to thousands of IT professionals since 2000.

Lawrence has developed and reviewed data warehouses for healthcare, telecommunications, engineering, broadcasting, financial services and public service organizations. He held the position of data warehouse practice leader at Linc Systems Corporation, CT, USA and vice-president of data warehousing products at Teleran Technologies, NJ, USA. Lawrence was also a Ralph Kimball Associate and has taught data warehousing classes for Kimball University in Europe and South Africa and contributed to Kimball articles and design tips. He lives in Yorkshire, England with his wife Mary and daughter Aimee. Lawrence can be contacted at:

lcorr@decisionone.co.uk

**Jim Stagnitto** is a data warehouse and master data management architect specializing in the healthcare, financial services and information service industries. He is the founder of the data warehousing and data mining consulting firm Llumino.

Jim has been a guest contributor for Ralph Kimball's Intelligent Enterprise column, and a contributing author to Ralph Kimball & Joe Caserta's *The Data Warehouse ETL Toolkit*. He lives in Bucks County, PA, USA with his wife Lori, and their happy brood of pets. Jim can be contacted at:

jim.stagnitto@llumino.com

# ACKNOWLEDGEMENTS

We would like to express our gratitude to everyone who made this book about BEAM✲ possible (using BEAM✲ notation):

**ACKNOWLEDGMENTS [RE]**

| AUTHOR | **would like to thank** PERSON | at PLACE | for VERY GOOD REASON |
|---|---|---|---|
| [who] MV, C | [who] MV, ML, C | [where] MV, ML, C | [why] T |
| Lawrence and Jim | Ralph and Julie Kimball | The Kimball Group | Dimensional modeling inspiration and much more |
| Lawrence | Pat Harrell, Dave Taylor, and Richard Synoradzki | Linc Systems | Collaborative modeling inspiration |
| Lawrence | Bob Hefner | Linc Systems | Career oportunities and the chance to meet all of the above |
| Lawrence | Alan Simpson | Standard Life | Fact-specific calendar technique |
| Lawrence | John Telford and Ian Raven | Channel 4 | Timelines and proactive design inspiration |
| Lawrence | Laurence Hesketh Geoff Hurrell | Northwards Fidelity | Eagle eyes |
| Lawrence | Mary and Aimee | Home | Peace, love and understanding |
| Lawrence | Ian Fleming John le Carré | In print, on screen | Entertainment, suspense |
| Jim | Joe Caserta | Caserta Concepts | Introduction to Lawrence, coaching |
| Jim | Lori | Home | Endless patience, love and support |
| Jim | Sadie the kitten | Home | Comic relief |
| Lawrence and Jim | Our clients | UK, USA, RSA | Numerous design challenges |
| Lawrence and Jim | Our business partners | UK, USA, RSA | Their hard work |
| Lawrence and Jim | Roger Thomas Bob Young | JCK Consulting Ideal Systems | Allowing us to adapt BEAM |
| Lawrence and Jim | Chris Adamson | Oakton Software | Author advice, dimensional modeling techniques |
| Lawrence and Jim | Rudyard Kipling | Any library, Project Gutenberg | Six honest serving-men, Baloo |

# CONTENTS

BEAM✲


✲ Business Event Analysis & Modeling

# INTRODUCTION

Dimensional modeling is responsible for today's DW/BI successes, yet we still struggle to deliver enough BI

Dimensional modeling, since it was first popularized by Ralph Kimball in the mid-1990s, has become the accepted (data modeling) technique for designing the high performance data warehouses that underpin the success of today's business intelligence (BI) applications. Yet, with an ever increasing number of BI initiatives stumbling long before they reach the data modeling phase, it has become clear that Data Warehousing/Business Intelligence (DW/BI) needs new techniques that can revolutionize BI requirements analysis in the same way that dimensional modeling has revolutionized BI database design.

Agile techniques can help, but they must address data warehouse design, not just BI application development

Agile, with its mantra of creating business value through the early and frequent delivery of working software and responding to change, has had just such a revolutionary effect on the world of application development. Can it take on the challenges of DW/BI? Agile's emphasis on collaboration and incremental development coupled with techniques such as Scrum and User Stories, will certainly improve *BI application development*—once a data warehouse is in place. But to truly have an impact on DW/BI, agile must also address *data warehouse design* itself. Unfortunately, the agile approaches that have emerged, so far, are vague and non-prescriptive in this one key area. For *agile BI* to be more than a marketing reboot of business-as-usual business intelligence, it must be *agile DW/BI* and we, DW/BI professionals, must do what every true agilist would recommend: adapt agile to meet our needs while still upholding its values and principles (see Appendix A: The Agile Manifesto). At the same time, agilists coming afresh to DW/BI, for their part, must learn our hard-won data lessons.

This book is about BEAM✲: an agile approach to dimensional modeling

With that aim in mind, this book introduces BEAM✲ (*Business Event Analysis & Modeling*): a set of collaborative techniques for *modelstorming* BI data requirements and translating them into dimensional models on an agile timescale. We call the BEAM✲ approach "modelstorming" because it combines data modeling and brainstorming techniques for rapidly creating inclusive, understandable models that fully engage BI stakeholders.

BEAM✲ is used for *modelstorming* BI requirements directly with BI stakeholders

BEAM✲ modelers achieve this by asking stakeholders to tell *data stories,* using the *7W* dimensional types—*who, what, when, where, how many, why,* and *how*—to describe the business events they need to measure. BEAM✲ models support modelstorming by differing radically from conventional entity-relationship (ER) based models. BEAM✲ uses tabular notation and example data stories to define business events in a format that is instantly recognizable to spreadsheet-literate BI stakeholders, yet easily translated into atomic-detailed star schemas. By doing so, BEAM✲ bridges the business-IT gap, creates consensus on data definitions and generates a sense of business ownership and pride in the resulting database design.

## Who Is This Book For?

This book is for the whole agile DW/BI team, to help you not only gather requirements but also communicate design ideas

This book is intended for data modelers, business analysts, data architects, and developers working on data warehouses and business intelligence systems. All members of an agile DW/BI team—not just those directly responsible for gathering BI requirements or designing the data warehouse—will find the BEAM✲ notation a powerful addition to standard entity-relationship diagrams for communicating dimensional design ideas and estimating data tasks with their colleagues. To get the most from this book, readers should have a basic knowledge of database concepts such as *tables*, *columns*, *rows*, *keys,* and *joins*.

It is aimed at both new and experienced DW/BI practitioners. It's a quick-study guide to dimensional modeling and a source of new dimensional design patterns

For those new to data warehousing, this book provides a quick-study introduction to dimensional modeling techniques. For those of you who would like more background on the techniques covered, the later chapters and Appendix C provide references to case studies in other texts that will help you gain additional business insight. Experienced data warehousing professionals will find that this book offers a fresh perspective on familiar dimensional modeling patterns, covering many in more detail than previously available, and adding several new ones. For all readers, this book offers a radically new agile way of engaging with business users and kick-starting their next warehouse development project.

## Meet The Modelstormers or How To Use This Book

*Hello, I'm over here and I'm your fast track through this book*

You may have already noticed the marginalia (non-contagious), on your left at the moment. This provides a "fast track" summary for readers in a hurry. This agile path through our text was inspired by David A. Taylor's object technology series of books. The margins of this book also contain a cast of anything but marginal characters. They are the *modelstormers* you need on your agile DW/BI team. We used them to highlight key features in the text such as *tips*, *warnings*, *references* and *example modeling dialogues*. They appear in the following order (in Chapters 1-9):

**The bright modeler**, not surprisingly, has some bright ideas. His tips, techniques and practical modeling advice, distilled from the current topic, will help you improve your design.

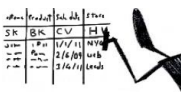
**The experienced dimensional modeler** has seen it all before. He's here to warn you when an activity or decision can steal your time, sanity or agility. Later in the book he follows the *pattern users* (see below) to tell you about the consequences or side effects of using their recommended design patterns. He would still recommend you use their patterns though—just with a little care.

**The note takers** are the members of the team who always read the full instructions before they use that new gadget or technique. They're always here to tell you to "make a note of that" when there is extra information on the current topic.


**The agilists** will let you know when we're being particularly agile. They wave their banner whenever a design technique supports a core value of the agile manifesto or principle of agile software development. These are listed in Appendix A.


**The modelstormers** appear en masse when we describe collaborative modeling and team planning, particularly when we offer practical advice and tips on using whiteboards and other inclusive tools for modelstorming.

**The scribe** appears whenever we introduce new BEAM✲ diagrams, notation conventions or short codes for rapidly documenting your designs. All the scribe's short codes are listed in Appendix B.



**The agile modeler** engages with stakeholders and facilitates modelstorming. She is here to ask example BEAM✲ questions, using the *7Ws*, to get stakeholders to tell their *data stories*.

**The stakeholders** are the subject matter experts, operational IT staff, BI users and BI consumers, who know the data sources, or know the data they want—anyone who can help define the data warehouse who is not a member of the DW/BI development team. They are here to provide example answers to the *agile modeler's* questions, tell data stories and pose their own tricky BI questions.

**The bookworm** points you to further reading on the current topic. All her reading recommendations are gathered in Appendix C.

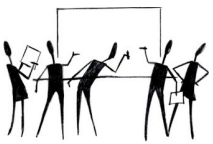
**The agile developer** appears when we have some practical advice about using software tools or there is something useful you can download.

**The head scratcher** has interesting/vexing DW/BI problems or requirements that the data warehouse design is going to have to address.

**The pattern users** have a solution to *the head scratcher's* problems. They're going to use tried and tested dimensional modeling design patterns, some new in print.

# How This Book Is Organized

This book has two parts. The first part covers agile dimensional modeling for BI data requirements gathering, while the second part covers dimensional design patterns for efficient and flexible star schema design.



Collaborative modeling with BI stakeholders

## Part I: Modelstorming

Part I describes how to modelstorm BI stakeholders' data requirements, validate these requirements using agile data profiling, review and prioritize them with stakeholders, estimate their ETL tasks as a team, and convert them into star schemas. It illustrates how agile data modeling can be used to replace traditional BI requirements gathering with accelerated database design, followed by BI prototyping to capture the real reporting and analysis requirements. Chapter 1 provides an introduction to dimensional modeling. Chapters 2 to 4 provide a step-by-step guide for using BEAM✲ to model business events and dimensions. Chapter 5 describes how BEAM✲ models are validated and translated into physical dimensional models and development sprint plans.

### Chapter 1: How to Model a Data Warehouse

Why we need new agile approaches for gathering BI requirements. Why they should be dimensional. What they should look like

**Data warehouses and operational systems:** Understanding the motivation for using dimensional modeling as the basis for agile database design.
**Dimensional modeling fundamentals:** Contrasting dimensional modeling with entity-relationship (ER) modeling, and learning the basic concepts and vocabulary of facts, dimensions, and star schemas that will be used throughout the book.
**Agile data modeling for analysis *and* design:** The BI requirement gathering problem. The challenges and opportunities of proactive DW/BI. The benefits of agile data warehousing. Why model with BI stakeholders? The case for modelstorming: using *agile* dimensional modeling to gather BI data requirements.
**Introduction to BEAM✲**: Comparison of BEAM✲ and ER diagrams.

### Chapter 2: Modeling Business Events

Step-by-step modeling of a business event using BEAM✲

**Discovering business events:** Using subjects, verbs, and objects to discover business events and tell *data stories*.
**Documenting business events:** Using whiteboards and spreadsheets and BEAM✲ tables to collaboratively model events.
**Discovering event details:** Using the *7Ws*: *who, what, when, where, how many, why,* and *how* to discover atomic-level event details. Using prepositions to connect details to events, and *data story themes* to define and document them. Using BEAM✲ *short codes* to document *event story types* (discrete, recurring, and evolving) and potential fact table granularity.

### Chapter 3: Modeling Business Dimensions

**Modeling *"detail about detail":*** Discovering dimensions and documenting their attributes with stakeholders. Telling *dimension stories* and overcoming weak narratives.

**Discovering dimensional hierarchies:** Using *hierarchy charts* to model hierarchical relationships and discover additional dimensional attributes.

**Documenting historical value requirements:** Using *change stories* and BEAM✲ short codes to define and document *slowly changing dimension* policies for supporting current (as is) and historically correct (as was) analysis views.

Step-by-step modeling of dimensions and hierarchies

### Chapter 4: Modeling Business Processes

**Modeling multiple business events:** Modelstorming with an *event matrix* to storyboard a data warehouse design by identifying and documenting the relationships between events and dimensions. Using event stories to prioritize requirements and plan development sprints.

**Modeling for agile data warehouse development:** Defining and reusing *conformed* dimensions. Generalizing dimensions and documenting their roles. Supporting incremental development and creating a *data warehouse bus architecture.*

Step-by-step modeling multiple business events and conformed dimensions

### Chapter 5: Modeling Star Schemas

**Agile data profiling:** Reviewing and adapting stakeholder models to data realities. Using BEAM✲ annotation to document data sources and physical data types, provide feedback to stakeholders on model viability and help estimate ETL tasks as a team.

**Converting BEAM✲ tables to star schemas:** defining and using surrogate keys to complete dimension tables, and convert event tables to fact tables. Using BEAM✲ technical codes to document the database design decisions and generate database schemas using the BEAM✲*Modelstormer* spreadsheet. Prototyping to define BI reporting requirements. Creating enhanced star schemas and physical dimensional matrices for a technical audience.

Validating stakeholder models and converting them into star schemas

## Part II: Dimensional Design Patterns


Collaborative modeling within the DW/BI team. Using design patterns associated with each of the *7W* dimensional types

Part II covers dimensional modeling techniques for designing high-performance star schemas. For this, we take a **design pattern approach** using a combination of BEAM✲ and star schema ER notation to capture significant DW/BI **requirements**, explain their associated issues/**problems,** and document pattern **solutions** and the **consequences** of implementing them. We have organized these design patterns around the *7W* dimensional types discovered in Part I. By using the *7Ws* to examine the complexities of modeling customers and employees (*who*), products and services (*what*), time (*when*), location (*where*), business measures (*how many*),

cause (*why*), and effect (*how*), we document new and established dimensional techniques from a *dimensional perspective* for the first time.

## Chapter 6: Who and What: People and Organizations, Products and Services

Design patterns for customer, employee and product dimensions

**Modeling customers, employees, and organizations:** Handling large, rapidly changing dimension populations. Tracking changes using *mini-dimensions*.

**Mixed business models:** Using *exclusive attributes* and *swappable dimensions* to model heterogeneous customers (businesses and consumer) and products (tangible goods and services).

**Advanced slowly changing Patterns:** Modeling micro and macro-level change. Supporting simultaneous current, historical, and previous value reporting requirements using *hybrid SCD views*.

**Representing complex hierarchical relationships:** Using *hierarchy maps* to handle recursive hierarchies, such as customer ownership, employee HR reporting structures, and product composition (component bill of materials and product bundles).

**Supporting variation within business events:** Using *multi-level dimensions* to describe events with variable granularity such as sales transactions assigned to individual employees or to teams, web advertisement impressions for single products or whole product categories.

## Chapter 7: When and Where: Time and Location

Design patterns for time and location dimensions

**Modeling time dimensionally:** Using separate calendar and clock dimensions and defining date keys.

**Year-to-date (YTD) analysis:** Using *fact state* tables and *fact-specific calendars* to support correct YTD comparisons.

**Time of day bracketing:** Designing custom business clocks that vary by day of week or time of year.

**Multinational calendars:** Modeling multinational dimensions that cope with time *and* location. Supporting time zones and national language reporting.

**Modeling movement:** Overloading events with additional time and location dimensions to understand journeys and trajectories.

## Chapter 8: How Many: Facts and Measures and KPIs

Design patterns for modeling efficient fact tables and flexible facts

**Designing fact tables for performance and ease of use:** Defining the three basic fact table patterns: *transactions*, *periodic snapshots*, and *accumulating snapshots*. Using event timelines to model accumulating snapshots as evolving events.

**Providing the basis for flexible measures and KPIs:** Defining atomic-level *additive facts*. Documenting *semi-additive* and *non-additive facts*, and understanding their limitations.

**Fact table performance optimization:** Using indexing, partitioning, and aggregation to improve fact table ETL and query performance.

**Cross-process analysis:** Combining the results from multiple fact tables using *drill-across* processing and *multi-pass* queries. Building *derived fact tables* and consolidated data marts to simplify query processing.

### Chapter 9: Why and How: Cause and Effect

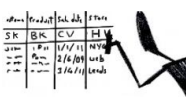
Design patterns for modeling cause and effect

**Modeling causal factors:** Using promotions, weather, and other causal dimensions to explain *why* events occur and *why* facts vary. Using text dimensions to handle unstructured reasons and exception descriptions.
**Modeling event descriptions:** Using *how* dimensions to collect any additional descriptions of an event. Consolidating excessive degenerate dimensions as *how* dimensions, and combining small *why* and *how* dimensions.
**Multi-valued dimensions**: Using *bridge tables* and weighting factors to handle fact allocation ('splitting the atom') when dimensions have multiple values for each atomic-level fact. Using *optional bridge tables* and *multi-level* dimensions to efficiently handle *barely* multi-valued dimensions. Using *pivoted* dimensions to support complex multi-valued constraints.
**Providing additional *how* dimensions:** Using *step* dimensions for understanding sequential behavior, *audit* dimensions for tracking data quality/lineage, and r*ange band* dimensions for treating facts as dimensions.

### Appendix A: The Agile Manifesto


Appendix A lists the four values of, and the twelve principles behind, *the manifesto for agile software development*.

### Appendix B: BEAM✲ Table Notation and Short Codes



Appendix B summarizes the BEAM✲ notation used throughout this book for modeling data requirements, recording data profiling results and representing physical dimensional modeling design decisions.

### Appendix C: Resources for Agile Dimensional Modelers

Appendix C lists books, websites, and tools (hardware and software) that will help you adopt and adapt the ideas contained in the book.

## Companion Website

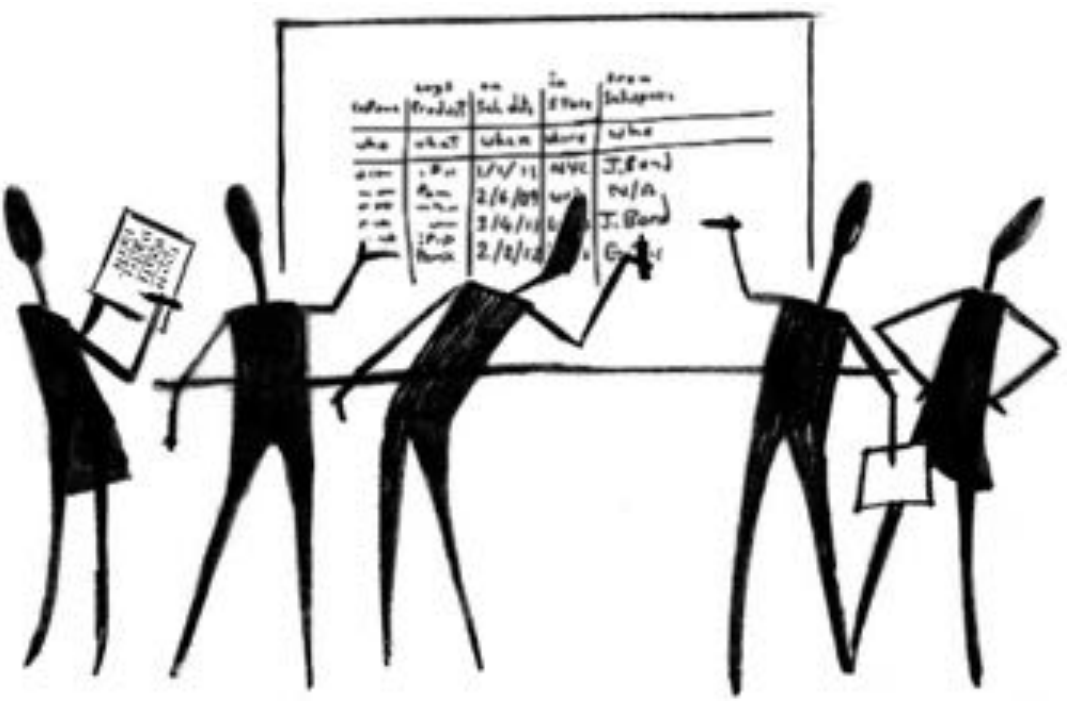
Visit **modelstorming.com** to download the BEAM✲*Modelstormer* spreadsheet and other templates that accompany this book. On the site you will find example models and code listings together with links to articles, books, and the worldwide schedule of training courses and workshops on BEAM✲ and agile data warehouse design. Register your paperback copy online to receive a discounted eBook version.

# PART I: MODELSTORMING

### AGILE DIMENSIONAL MODELING, FROM WHITEBOARD TO STAR SCHEMA

Dimensional Modeling: it's too important to be left to data modelers alone

*— Anon.*




# 1

# How to Model a Data Warehouse

Essentially, all models are wrong, but some are useful.
*— George E. P. Box*

Dimensional modeling supports data warehouse *design*

In this first chapter we set out the motivation for adopting an agile approach to data warehouse design. We start by summarizing the fundamental differences between data warehouses and online transaction processing (OLTP) databases to show why they need to be *designed* using very different data modeling techniques. We then contrast entity-relationship and dimensional modeling and explain why dimensional models are optimal for data warehousing/business intelligence (DW/BI). While doing so we also describe how dimensional modeling enables incremental design and delivery: key principles of agile software development.

*Collaborative* dimensional modeling supports *agile* data warehouse *analysis* and design

Readers who are familiar with the benefits of *traditional* dimensional modeling may wish to skip to **Data Warehouse Analysis and Design** on Page 11 where we begin the case for *agile* dimensional modeling. There, we take a step back in the DW/BI development lifecycle and examine the traditional approaches to data requirements *analysis*, and highlight their shortcomings in dealing with ever more complex data sources and aggressive BI delivery schedules. We then describe how *agile data modeling* can significantly improve matters by actively involving business stakeholders in the analysis *and* design process. We finish by introducing BEAM✲ (Business Event Analysis and Modeling): the set of agile techniques for *collaborative* dimensional modeling described throughout this book.

**Chapter 1 Topics**
***At a Glance***

- **Differences between operational systems and data warehouses**
- **Entity-relationship (ER) modeling vs. dimensional modeling**
- **Data-driven analysis and reporting requirements analysis limitations**
- **Proactive data warehouse design challenges**
- **Introduction to BEAM✲: an agile dimensional modeling method**

# OLTP vs. DW/BI: Two Different Worlds

OLTP and DW/BI have radically different DBMS requirements

Operational systems and data warehouses have fundamentally different purposes. Operational systems support the *execution* of business processes, while data warehouses support the *evaluation* of business processes. To execute efficiently, operational systems must be optimized for online transaction processing (OLTP). In contrast, data warehouses, must be optimized for query processing and ease of use. Table 1-1 highlights the very different usage patterns and database management system (DBMS) demands of the two types of system.

**Table 1-1** Comparison between OLTP databases and Data Warehouses

| CRITERIA | OLTP DATABASE | DATA WAREHOUSE |
|---|---|---|
| **Purpose** | Execute individual business processes ("turning the handles") | Evaluate multiple business processes ("watching the wheels turn") |
| **Transaction type** | Insert, select, update, delete | Select |
| **Transaction style** | Predefined: predictable, stable | Ad-hoc: unpredictable, volatile |
| **Optimized for** | Update efficiency and write consistency | Query performance and usability |
| **Update frequency** | Real-time: when business events occur | Periodic, (daily) via scheduled ETL (extract, transform, load). Moving to near real-time |
| **Update concurrency** | High | Low |
| **Historical data access** | Current and recent periods | Current + several years of history |
| **Selection criteria** | Precise, narrow | Fuzzy, broad |
| **Comparisons** | Infrequent | Frequent |
| **Query complexity** | Low | High |
| **Tables/joins per transaction** | Few (1–3) | Many (10+) |
| **Rows per transaction** | Tens | Millions |
| **Transactions per day** | Millions | Thousands |
| **Data volumes** | Gigabytes–Terabytes | Terabytes–Petabytes (many sources, history) |
| **Data** | Mainly raw detailed data | Detailed data, summarized data, derived data |
| **Design technique** | Entity-Relationship modeling (normalization) | Dimensional modeling |
| **Data model diagram** | ER diagram | Star schema |

# The Case Against Entity-Relationship Modeling

ER modeling is used to design OLTP databases

Entity-Relationship (ER) modeling is the standard approach to data modeling for OLTP database design. It classifies all data as one of three things: an entity, a relationship, or an attribute. Figure 1-1 shows an example entity-level ER diagram (ERD). Entities are shown as boxes and relationships as lines linking the boxes. The *cardinality* of each relationship—the number of possible matching values on either side of the relationship—is shown using *crow's feet* for many, | for one, and O for zero (also known as *optionality*).

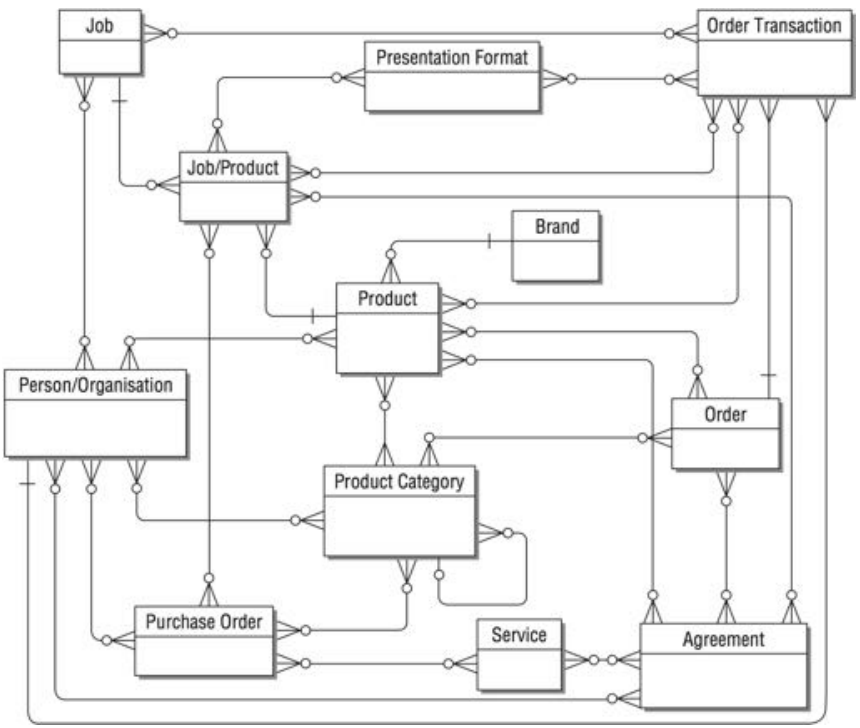



**Figure 1-1**
Entity-Relationship diagram (ERD)

Entities become tables, attributes become columns

Within a relational database, entities are implemented as tables and their attributes as columns. Relationships are implemented either as columns within existing tables or as additional tables depending on their cardinality. One-to-one (1:1) and many-to-one (M:1) relationships are implemented as columns, whereas many-to-many (M:M) relationships are implemented using additional tables, creating additional M:1 relationships.

ER models are typically in third normal form (3NF)

ER modeling is associated with normalization in general, and *third normal form* (3NF) in particular. ER modeling and normalization have very specific technical goals: to reduce data redundancy and make explicit the 1:1 and M:1 relationships within the data that can be enforced by relational database management systems.

## Advantages of ER Modeling for OLTP

3NF is efficient for transaction processing

Normalized databases with few, if any, data redundancies have one huge advantage for OLTP: they make write transactions (inserts, updates, and deletes) very efficient. By removing data redundancies, transactions are kept as small and simple as possible. For example, the repeat usage of a service by a telecom's customer is recorded using tiny references to the customer and service: no unnecessary details are rerecorded each time. When a customer or service detail changes (typically) only a single row in a single table needs to be updated. This helps avoid update anomalies that would otherwise leave a database in an inconsistent state.

**Higher forms of normalization are available, but most ER modelers are satisfied when their models are in 3NF. There is even a mnemonic to remind everyone that data in 3NF depends on "The key, the whole key, and nothing but the key, so help me Codd"—in memory of Edgar (Ted) Codd, inventor of the relational model.**

## Disadvantages of ER Modeling for Data Warehousing

3NF is inefficient for query processing

Even though 3NF makes it easier to get data in, it has a huge disadvantage for BI and data warehousing: it makes it harder to get the data out. Normalization proliferates tables and join paths making queries (SQL selects) less efficient and harder to code correctly. For example, looking at the Figure 1-1 ERD, could you estimate how many ways PRODUCT CATEGORY can be joined to ORDER TRANSACTION? A physical 3NF version of the model would contain at least 20 more tables to resolve the M:M relationships. Faced with such 3NF databases, even the simplest BI query requires multiple tables to be joined through multiple intermediate tables. These long joins paths are difficult to optimize and queries invariably run slowly.

3NF models are difficult to understand

More importantly, queries will only produce the right answers if users navigate the right join paths, i.e., ask the right questions in SQL terms. If the wrong joins are used, they unknowingly get answers to some other (potentially meaningless) questions. 3NF models are complex for both people and machines. Specialist hardware (data warehouse appliances) is improving query/join performance all the time, but the human problems are far more difficult to solve. Smart BI software can hide database schema complexity behind a semantic layer, but that merely moves the burden of understanding a 3NF model from BI users at query time to BI developers at configuration time. That's a good move but its not enough. 3NF models remain too complex for business stakeholders to review and quality assure (QA).

History further complicates 3NF

ER models are further complicated by data warehousing requirements to track history in full to support valid 'like-for-like' comparisons over time. Providing a true historical perspective of business events requires that many otherwise simple descriptive attributes become time relationships, i.e., existing M:1 relationships become M:M relationships that translate into even more physical tables and com-

plex join paths. Such temporal database designs can defeat even the smartest BI tools and developers.

Laying out a readable ERD for any non-trivial data model isn't easy. The mnemonic "dead crows fly east" encourages modelers to keep crows' feet pointing up or to the left. Theoretically this should keep the high-volume volatile entities (transactions) top left and the low-volume stable entities (lookup tables) bottom right. However, this layout seldom survives as modelers attempt to increase readability by moving closely related or commonly used entities together. The task rapidly descends into an exercise in trying to reduce overlapping lines. Most ERDs are visually overwhelming for BI stakeholders and developers who need simpler, human-scale diagrams to aid their communication and understanding.

Large readable ER diagrams are difficult to draw: all those overlapping lines

## The Case For Dimensional Modeling

Dimensional models define business processes and their individual events in terms of measurements (*facts*) and descriptions (*dimensions*), which can be used to filter, group, and aggregate the measurements. Data cubes are often used to visualize simple dimensional models, as in Figure 1-2, which shows the multidimensional analysis of a sales process with three dimensions: PRODUCT (*what*), TIME (*when*), and LOCATION (*where*). At the intersection of these dimensional values there are interesting facts such as the quantity sold, sales revenue, and sales costs. This perspective on the data appeals to many BI users because the three-dimensional cube can be thought of as a stack of two-dimensional spreadsheets. For example, one spreadsheet for each location contains rows for products, columns for time periods, and revenue figures in each cell.

Dimensional models appeal to spreadsheet-savvy BI users

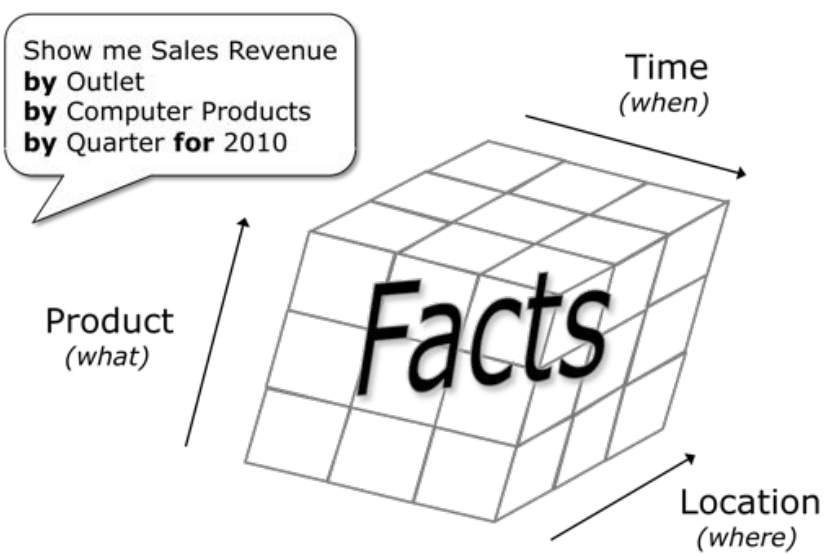



**Figure 1-2**
Multidimensional analysis

## Star Schemas

Star schemas are used to visualize dimensional models

Real-world dimensional models are used to measure far more complex business processes (with more dimensions) in far greater detail than could be attempted using spreadsheets. While it is difficult to envision models with more than three dimensions as multi-dimensional cubes (they wouldn't actually be cubes), they can easily be represented using *star schema* diagrams. Figure 1-3 shows a classic star schema for retail sales containing a fourth (causal) dimension: PROMOTION, in addition to the dimensional attributes and facts from the previous cube example.

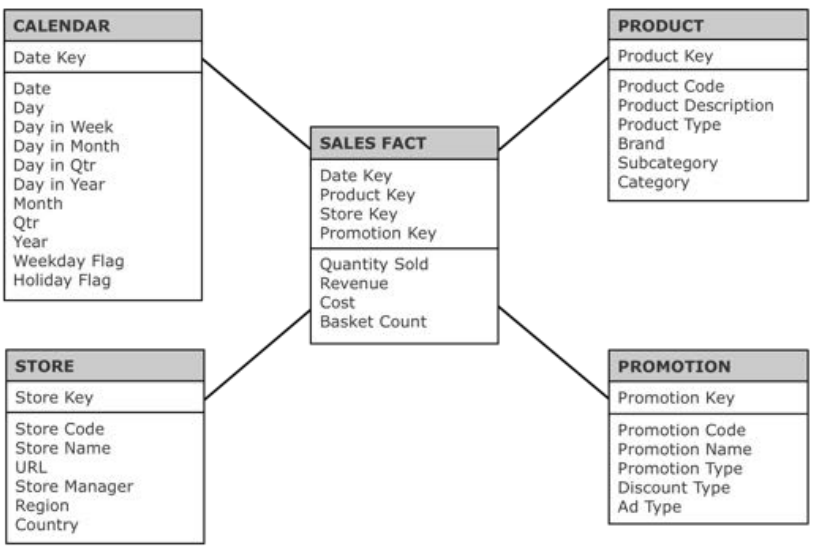



**Figure 1-3**
Sales star schema


***Star schema* is also the term used to describe the physical implementation of a dimensional model as relational tables.**

Star schema diagrams are non-normalized (N3NF) ER representations of dimensional models. When drawn in a database modeling tool they can be used to generate the SQL for creating fact and dimension tables in relational database management systems. Star schemas are also used to document and define the data cubes of multidimensional databases.

**ER diagrams work best for viewing a small number of tables at one time. How many tables? About as many as in a dimensional model: a star schema.**

## Fact and Dimension Tables

Star schemas are comprised of fact and dimension tables

A star schema is comprised of a central fact table surrounded by a number of dimension tables. The fact table contains facts: the numeric (quantitative) measures of a business event. The dimension tables contain mainly textual (qualitative) descriptions of the event and provide the context for the measures. The fact table also contains dimensional foreign keys; to an ER modeler it represents a M:M

relationship between the dimensions. A subset of the dimensional foreign keys form a composite primary key for the fact table and defines its *granularity,* or level of detail.


**The term *dimension* in this book refers to a dimension table whereas *dimensional attribute* refers to a column in a dimension table.**

Dimensional hierarchies support drill-down analysis

Dimensions contain sets of descriptive (dimensional) attributes that are used to filter data and group facts for aggregation. Their role is to provide good report row headers and title/heading/footnote filter descriptions. Dimensional attributes often have a hierarchical relationship that allows BI tools to provide drill-down analysis. For example, drilling down from Quarter to Month, Country to Store, and Category to Product.

Dimensions are small, fact tables are large

Not all dimensional attributes are text. Dimensions can contain numbers and dates too, but these are generally used like the textual attributes to filter and group the facts rather than to calculate aggregate measures. Despite their width, dimensions are tiny relative to fact tables. Most dimensions contain considerably less than a million rows.

**The most useful facts are additive measures that can be aggregated using any combination of the available dimensions. The most useful dimensions provide rich sets of descriptive attributes that are familiar to BI users.**

## Advantages of Dimensional Modeling for Data Warehousing

Dimensional models maximize query performance and usability

The most obvious advantage of a dimensional model, noticeable in Figure 1-3, is its *simplicity*. The small number of tables and joins, coupled with the explicit facts in the center of the diagram, makes it easy to think about how sales can be measured and easy to construct the necessary queries. For example, if BI users want to explore product sales by store, only one short join path exists between PRODUCT and STORE: through the SALES FACT table. Limiting the number of tables involved and the length of the join paths in this way maximizes query performance by leveraging DBMS features such as star-join optimization (which processes multiple joins to a fact table in a single pass).

Dimensional models are process-oriented. They represent business processes described using the *7Ws* framework

A deeper, less immediately obvious benefit of dimensional models is that they are *process-oriented*. They are not just the result of some aggressive physical data model optimization (that has denormalized a logical 3NF ER model into a smaller number of tables) to overcome the limitations of databases to cope with join intensive BI queries. Instead, the best dimensional models are the result of asking questions to discover which business processes need to be measured, how they should be described in business terms and how they should be measured. The resulting dimensions and fact tables are not arbitrary collections of denormalized data but *the 7Ws* that describe the full details of each individual business event worth measuring.

# The *7Ws* Framework

***Who*** is involved?
***What*** did they do? To ***what*** is it done?
***When*** did it happen?
***Where*** did it take place?
***HoW many*** or much was recorded – ***how*** can it be measured?
***Why*** did it happen?
***HoW*** did it happen – in what manner?

The *7Ws* are interrogatives: question forming words

The *7Ws* are an extension of the 5 or 6Ws that are often cited as *the* checklist in essay writing and investigative journalism for getting the 'full' story. Each W is an *interrogative*: a word or phrase used to make questions. The *7Ws* are especially useful for data warehouse data modeling because they focus the design on BI activity: asking questions.


**Fact tables represent *verbs* (they record business process *activity*). The facts they contain and the dimensions that surround them are *nouns*, each classifiable as one of the *7Ws*. *6Ws*: *who, what, when, where, why*, and *how* represent dimension types. The 7th W: *how many*, represents facts. BEAM✻ *data stories* use the *7Ws* to discover these important verb and noun combinations.**

Star schemas usually contain 8-20 dimensions

Detailed dimensional models usually contain more than 6 dimensions because any of the *6Ws* can appear multiple times. For example, an order fulfillment process could be modeled with 3 *who* dimensions: CUSTOMER, EMPLOYEE, and CARRIER, and 2 *when* dimensions: ORDER DATE and DELIVERY DATE. Having said that, most dimensional models do not have many more than 10 or 12 dimensions. Even the most complex business events rarely have 20 dimensions.

Star schemas support agile, incremental BI

The deep benefit of process-oriented dimensional modeling is that it naturally breaks data warehouse scope, design and development into manageable chunks consisting of just the individual business processes that need to be measured next. Modeling each business process as a separate star schema supports incremental design, development and usage. Agile dimensional modelers and BI stakeholders can concentrate on one business process at a time to fully understand how it should be measured. Agile development teams can build and incrementally deliver individual star schemas earlier than monolithic designs. Agile BI users can gain early value by analyzing these business processes initially in isolation and then grow into more valuable, sophisticated cross-process analysis. Why develop ten stars when one or two can be delivered far sooner with less investment 'at risk'?


**Dimensional modeling provides a well-defined unit of delivery—the star schema—which supports the agile principles: *"Satisfy the customer through early and continuous delivery of valuable software."* and "*Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.*"**

# Data Warehouse Analysis and Design

Both 3NF ER modeling and dimensional modeling are primarily database *design* techniques (one arguably more suited to data warehouse design than the other). Prior to using either to design data structures for meeting BI information requirements, some form of *analysis* is required to discover these requirements. The two approaches commonly used to obtain data warehousing requirements are data-driven analysis (also known as supply driven) and reporting-driven analysis (also known as demand driven). While most modern data warehousing initiatives use some combination of the two, Figure 1-4 shows the analysis and design bias of early 3NF enterprise data warehouses compared to that of more recent dimensional data warehouses and data marts.

Analysis techniques are required to discover BI data requirements

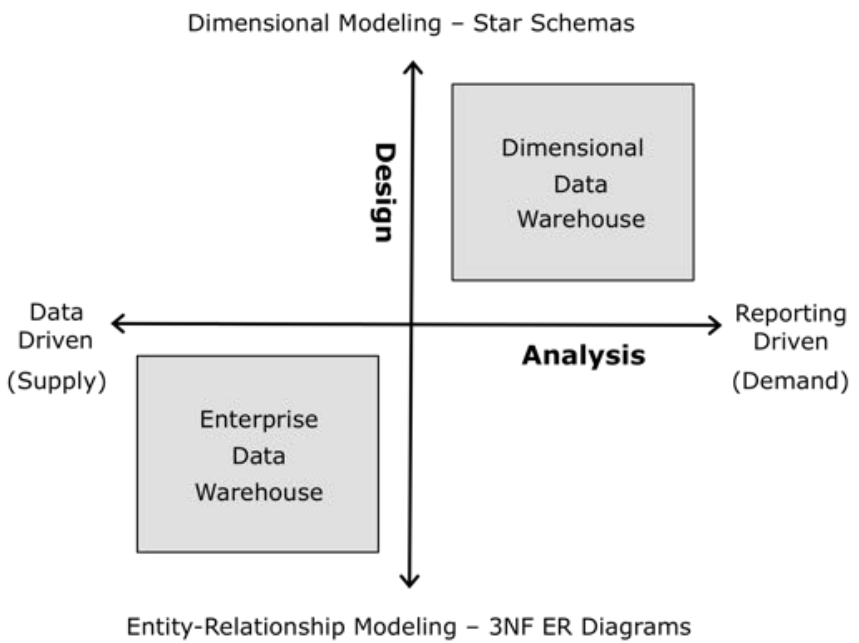



**Figure 1-4**
Data warehouse analysis and design biases

## Data-Driven Analysis

Using a data-driven approach, data requirements are obtained by analyzing operational data sources. This form of analysis was adopted by many early IT-lead data warehousing initiatives to the exclusion of all others. User involvement was avoided as it was mistakenly felt that data warehouse design was simply a matter of re-modeling multiple data sources using ER techniques to produce a single 'perfect' 3NF model. Only after that was built, would it then be time to approach the users for their BI requirements.

Pure data-driven analysis avoided early user involvement

Leading to DW designs that did not met BI user needs

Unfortunately, without user input to prioritize data requirements and set a manageable scope, these early data warehouse designs were time-consuming and expensive to build. Also, being heavily influenced by the OLTP perspective of the source data, they were difficult to query and rarely answered the most pressing business questions. Pure data-driven analysis and design became known as the "build it and they will come" or "field of dreams" approach, and eventually died out to be replaced by hybrid methods that included user requirements analysis, source data profiling, and dimensional modeling.

Packaged apps are especially challenging data sources to analyze

Data-driven analysis has benefited greatly from the use of modern data profiling tools and methods but despite their availability, data-driven analysis has become increasing problematic as operational data models have grown in complexity. This is especially true where the operational systems are packaged applications, such as Enterprise Resource Planning (ERP) systems built on highly generic data models.

IT staff are comfortable with data-driven analysis

In spite of its problems, data-driven analysis continues to be a major source of data requirements for many data warehousing projects because it falls well within the technical comfort zone of IT staff who would rather not get too involved with business stakeholders and BI users.

## Reporting-Driven Analysis

Reporting requirements are gathered by interviewing potential BI users in small groups

Using a reporting-driven approach, data requirements are obtained by analyzing the BI users' reporting requirements. These requirements are gathered by interviewing stakeholders one at a time or in small groups. Following rounds of meetings, analyst's interview notes and detailed report definitions (typically spreadsheet or word processor mock-ups) are cross-referenced to produce a consolidated list of data requirements that are verified against available data sources. The results requirements documentation is then presented to the stakeholders for ratification. After they have signed off the requirements, the documentation is eventually used to drive the data modeling process and subsequent BI development.

User involvement helps to create more successful DWs

Reporting-driven analysis focuses the data warehouse design on efficiently prioritizing the stakeholder's most urgent reporting requirements and *can* lead to timely, successful deployments when the scope is managed carefully.

Accretive BI reporting requirements are impossible to capture in full, in advance

Unfortunately, reporting-driven analysis is not without its problems. It is time-consuming to interview enough people to gather *'all'* the reporting requirements needed to attain an enterprise or even a cross-departmental perspective. Getting stakeholders to think beyond 'the next set of reports' and describe longer term requirements in sufficient detail takes considerable interviewing skills. Even experienced business analysts with generous requirement gathering budgets struggle because detailed analytical requirements by their very nature are *accretive*: they gradually build up layer upon layer. BI users find it difficult to articulate

future information needs beyond the 'next reports', because these needs are dependent upon the answers the 'next reports' will provide, and the unexpected new business initiatives those answers will trigger. The ensuing steps of collating requirements, feeding them back to business stakeholders, gaining consensus on data terms, and obtaining sign off can also be an extremely lengthy process.

Over-reliance on reporting requirements has lead to many initially successful data warehouse designs that fail to handle change in the longer-term. This typically occurs when inexperienced dimensional modelers produce designs that match the current report requests *too* closely, rather than treating these reports as clues to discovering the *underlying business processes* that should be modeled in greater detail to provide true BI flexibility. The problem is often exasperated by initial requirement analysis taking so long that there isn't the budget or willpower to swiftly iterate and discover the real BI requirements as they evolve. The resulting inflexible designs have led some industry pundits to unfairly brand dimensional modeling as too *report-centric*, suitable at the data mart level for satisfying the current reporting needs of individual departments, but unsuitable for enterprise data warehouse design. This is sadly misleading because dimensional modeling has no such limitation when used correctly to *iteratively* and *incrementally* model *atomic-level detailed* business processes rather than reverse engineer *summary* report requests.

Focusing too closely on current reports alone leads to inflexible dimensional models

## Proactive DW/BI Analysis and Design

Historically, data warehousing has lagged behind OLTP development (in technology as well as chronology). Data warehouses were built often long after well established operational systems were found to be inadequate for reporting purposes, and significant BI backlogs had built up. This *reactive* approach is illustrated on the example timeline in Figure 1-5.

Early DWs were reactive to OLTP reporting problems

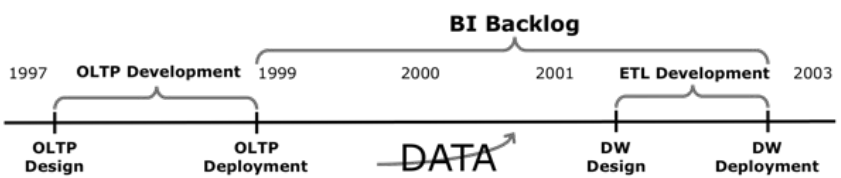



**Figure 1-5**
Reactive DW timeline

Today, DW/BI has caught up and become *proactive.* The two different worlds of OLTP and DW/BI have become parallel worlds where many new data warehouses need to go live/be developed concurrently with their new operational source systems, as shown on the Figure 1-6 timeline.

The lag between OLTP and DW roll-out is disappearing

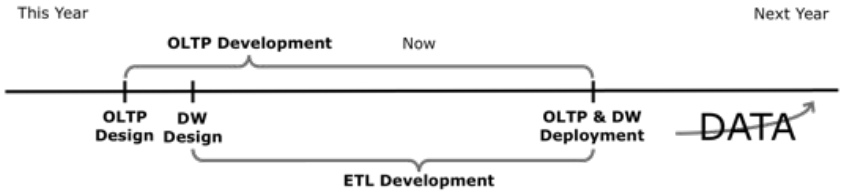



**Figure 1-6**
Proactive DW timeline

Proactive DW/BI addresses operational demands, avoids interim solutions and preempts BI performance problems

DW/BI has steadily become proactive for a number of business-led reasons:

- DW/BI itself has become more operational. The (largely technical) distinction between operational and analytical reporting has blurred. Increasingly, sophisticated operational processes are leveraging the power of (near real-time) BI and stakeholders want a one-stop shop for all reporting needs: the data warehouse.
- Organizations (especially those that already have DW/BI success) now realize that, sooner rather than later, each major new operational system will need its own data mart or need to be integrated with an existing data warehouse.
- BI stakeholders simply don't want to support 'less than perfect' interim reporting solutions and suffer BI backlogs.

## Benefits of Proactive Design for Data Warehousing

Proactive DW design can improve the data available for BI

When data warehouse design preempts detailed operational data modeling it can help BI stakeholders set the data agenda, i.e., stipulate their ideal information requirements whilst the new OLTP system is still in development and enhancements can easily be incorporated. This is especially significant for the definition of mandatory data. Vital BI attributes that might have been viewed as optional or insignificant from a purely operational perspective can be specified as not null and captured from day one—before operational users develop bad habits that might have them (inadvertently) circumvent the same enhancements made later. Agile OLTP development teams should welcome these 'early arriving changes'.

Proactive DW design can streamline ETL change data capture

ETL processes are often thought of as difficult/impossible to develop without access to stable data sources. However, when a data source hasn't been defined or is still a moving target, it gives the agile ETL team the chance to define its 'perfect' data extraction interface specification based on the proactive data warehouse model, and pass that on to the OLTP development team. This is a great opportunity for ETL designers to ensure that adequate *change data capture* functionality (e.g. consistently maintained timestamps and update reason codes) are built into all data sources so that ETL processes can easily detect when data has changed *and* for what reason: whether genuine change has occurred to previously correct values (that must be tracked historically) or mistakes have been corrected (which need no history).

**When source database schemas are not yet available, ETL development can still proceed if ETL and OLTP designers can agree on flat file data extracts. Once OLTP have committed to provide the specified extracts on a schedule to meet BI needs, ETL transformation and load routines can be developed to match this source to the proactive data warehouse design target.**

## Challenges of Proactive Analysis for Data Warehousing

Proactive analysis takes place before data exists

While being proactive has great potential benefits for DW/BI, the late appearance of *data* on the Figure 1-6 timeline unfortunately heralds further analysis challenges for data warehouse designers: BI requirements gathering must take place before any real data is available. Under these circumstances proactive data modelers can rely even less upon traditional analysis techniques to provide BI data requirements to match their aggressive schedule.

### Proactive Reporting-Driven Analysis Challenges

Reporting-driven analysis is difficult before data exists

Traditional interviewing techniques for gathering reporting requirements are problematic when stakeholders haven't seen the data or applications that will fuel their BI imagination. With no existing reports to work from, business analysts can't ask their preferred icebreaker question: "How can your favorite reports be improved?" and they have nothing to point at if and ask: "How do you use this data to make decisions?". Even more open questions such as "What decisions do you make and what information will help you to make them quicker/better?" can fall flat when a new operational systems will shortly enable an entirely new business process that stakeholders have no prior experience of measuring, or managing.

### Proactive Data-Driven Analysis Challenges

Data-driven analysis is impossible with no data to profile

IT cannot fall back on data-driven analysis: data profiling tools and database remodeling skills are of little use when new source databases don't exist, are still under development, or contain little or no representative data (only test data). Even when new operational systems are implemented using package applications with stable, (well) documented database schemas they are often too complicated for untargeted data profiling: it would take too long and be of little value if only a small percentage of the database is currently used/populated and well understood by the available IT resources.

### Data then Requirements: a 'Chicken or the egg' Conundrum

Proactive DW design requires a new approach to data analysis, modeling and design

Before there is data and users have lived with it for a time (with less than perfect BI access) both IT and business stakeholders cannot define genuine BI requirements in sufficient detail. Without these early detailed requirements proactive data warehouse designs routinely fail to provide the right information on time to avoid a BI backlog building up as soon as data is available. To solve this 'data then requirements'/'chicken or the egg' conundrum, proactive data warehousing needs a new approach to database analysis and design: not your father's data modeling, not even your father's dimensional modeling!

# Agile Data Warehouse Design

Traditional data warehousing follows a near-serial or *waterfall* approach to design and development

Traditional data warehousing projects follow some variant of *waterfall* development as summarized on the Figure 1-7 timeline. The shape of this timeline and the term 'waterfall' might suggest that its 'all downhill' after enough detailed requirements have been gathered to complete the 'Big Design Up Front' (BDUF). Unfortunately for DW/BI, this approach relies on a preternatural ability to exhaustively capture requirements upfront. It also postpones all data access and the hoped for BI value it brings until the (bitter) end of the waterfall (or rainbow!). For these reasons *pure* waterfall (analyze only once, design only once, develop only once, etc.) DW/BI development, whether by design or practice, is rare.

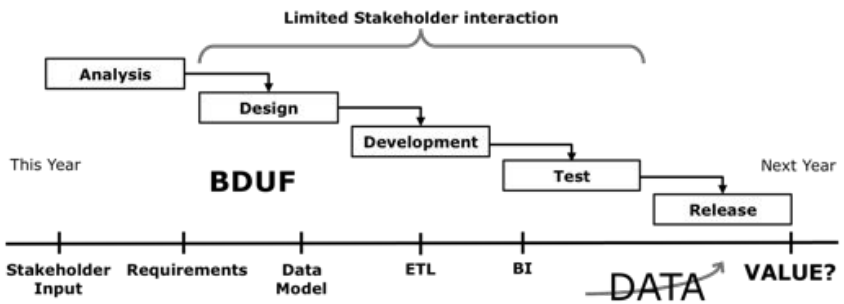



**Figure 1-7**
Waterfall DW development timeline

Dimensional modeling enables incremental development

Dimensional modeling can help reduce the risks of pure waterfall by allowing developers to release early incremental BI functionality one star schema at a time, get feedback and make adjustments. But even dimensional modeling, like most other forms of data modeling, takes a (near) serial approach to analysis and design (with 'Big Requirements Up Front' (BRUF) preceding BDUF data modeling) that is subject to the inherent limitations and initial delays described already.

Agile data warehousing is highly iterative and collaborative

Agile data warehousing seeks to further reduce the risks associated with upfront analysis and provide even more timely BI value by taking a highly iterative, incremental and collaborative approach to all aspects of DW design and development as shown on the Figure 1-8 timeline.

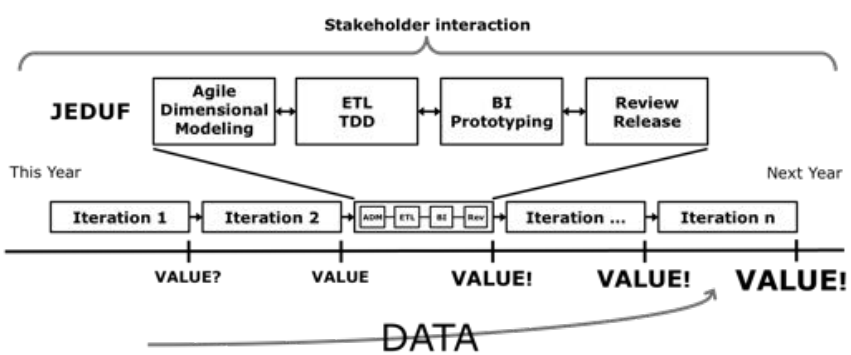



**Figure 1-8**
Agile DW development timeline

By avoiding the BDUF and instead doing 'Just Enough Design Upfront' (JEDUF) in the initial iterations and 'Just-In-Time' (JIT) detailed design within each iteration, agile development concentrates on the early and frequent delivery of working software that adds value, rather than the production of exhaustive requirements and design documentation that describes what will be done in the future to add value.

Agile focuses on the early and frequent delivery of working software that adds value

For agile DW/BI, the working software that adds value is a combination of queryable database schemas, ETL processes and BI reports/dashboards. The minimum set of valuable working software that can be delivered per iteration is a star schema, the ETL processes that populates it and a BI tool or application configured to access it. The minimum amount of design is a star.

For DW design, the minimum valuable working software is a star schema

To design any type of significant database schema to match the early and frequent delivery schedule of an agile timeline requires an equally agile alternative to the traditionally serial tasks of data requirements analysis and data modeling.

Agile database development needs agile data modeling

## Agile Data Modeling

Scott Ambler, author of several books on agile modeling and agile database techniques (www.agiledata.org) defines *agile data modeling* as follows: *"Data modeling is the act of exploring data-oriented structures. Evolutionary data modeling is data modeling performed in an iterative and incremental manner. Agile data modeling is evolutionary data modeling done in a collaborative manner."*

Agile data modeling is collaborative and evolutionary

*Iterative*, *incremental* and *collaborative* all have very specific meanings in an agile development context that bring with them significant benefits:

Collaborative modeling combines analysis and design and actively involves stakeholders

- **Collaborative data modeling** obtains data requirements by modeling directly with stakeholders. It effectively combines analysis and design and 'cuts to the chase' of producing a data model (working software and documentation) rather than 'the establishing shot' of recording data requirements (only documentation).

Evolutionary modeling supports incremental development by capturing requirements when they grow and change

- **Incremental data modeling** gives you more data requirements when they are better understood/needed by stakeholders, and when you are ready to implement them. Incremental modeling and development are scheduling strategies that support early and frequent software delivery.

- **Iterative data modeling** helps you to understand existing data requirements better and improve existing database schemas through refactoring: correcting mistakes and adding missing attributes which have now become available or important. Iterative modeling and development are rework strategies that increase software value.

# Agile Dimensional Modeling

DW/BI benefits from agile dimensional modeling

By taking advantage of dimensional modeling's unit of discovery—a business process worth measuring—agile data modeling has arguably greater benefits for DW/BI than any other type of database project:

Agile dimensional modeling focuses on business processes rather than reports

- Agile modeling avoids the 'analysis paralysis' caused by trying to discover the 'right' reports amongst the large (potentially infinite?) number of volatile, constantly re-prioritized requests in the BI backlog. Instead, agile dimensional modeling gets everyone to focus on the far smaller (finite) number of relatively stable business processes that stakeholders want to measure now or next.

Agile dimensional modeling creates flexible, report-neutral designs

- Agile dimensional modeling avoids the need to decode detailed business events from current summary report definitions. Modeling business processes without the blinkers of specific report requests produces more flexible, report-neutral, enterprise-wide data warehouse designs.

Agile modeling enables proactive DW/BI to influence operational system development

- Agile data modeling can break the "data then requirements" stalemate that exists for DW/BI just before a new operational system is implemented. Proactive agile dimensional modeling enables BI stakeholders to define new business processes from a measurement perspective and provide timely BI input to operational application development or package configuration.

Evolutionary modeling supports accretive BI requirements

- Agile modeling's evolutionary approach matches the accretive nature of genuine BI requirements. By following hands-on BI prototyping and/or real BI usage, iterative and incremental dimensional modeling allows stakeholders to (re)define their real data requirements.

Collaborative modeling teaches stakeholders to think dimensionally

- Many of the stakeholders involved in collaborative modeling will become direct users of the finished dimensional data models. Doing some form of dimensional modeling with these future BI users is an opportunity to teach them to think dimensionally about their data and define common, conformed dimensions and facts from the outset.

Collaborative modeling creates stakeholder pride in the data warehouse

- Collaborative modeling fully engages stakeholders in the design process, making them far more enthusiastic about the resultant data warehouse. It becomes *their* data warehouse, they feel invested in the data model and don't need to be trained to understand what it means. It contains their consensus on data terms because it is designed directly by them: groups of relevant business experts rather than the distillation of many individual report requests interpreted by the IT department.

**Never underestimate the affection stakeholders will have for data models that they *themselves* (help) create.**

## Agile Dimensional Modeling and Traditional DW/BI Analysis

Agile dimensional modeling doesn't completely replace traditional DW/BI analysis tasks, but by preceding both data-driven and reporting-driven analysis it can make them agile too: significantly reducing the work involved while improving the quality and value of the results.

Agile dimensional modeling makes traditional analysis tasks agile

### Agile Data-Driven Analysis

Agile data-driven analysis is streamlined by *targeted data profiling*. Only the data sources implicated by the agile data model need to be analyzed within each iteration. This targeted profiling supports the agile practice of *test-driven development* (TDD) by identifying the data sources that will be used to test the data warehouse design and ETL processes ahead of any detailed physical data modeling. If an ETL test can't be defined because a source isn't viable, agile data modelers don't waste time physically modeling what can't be tested, unless they are doing proactive data warehouse design. In this case the agile data warehouse model can assist the test-driven development of the new OLTP system.

Data-driven analysis becomes *targeted data profiling*

### Agile Reporting-Driven Analysis

Agile reporting-driven analysis takes the form of BI prototyping. The early delivery of dimensional database schemas enables the early extraction, transformations and loading (ETL) of real sample data so that better report requirements can be prototyped using the BI user's actual BI toolset rather than mocked-up with spreadsheets or word processors. It is intrinsically fairer to ask users to define their requirements and developers to commit to them, once everyone has a sense of what their BI tools are capable of, given the available data.

Reporting-driven analysis becomes BI prototyping

## Requirements for Agile Dimensional Modeling

Agile modeling requires both IT and business stakeholders to change their work practices and adopt new tools and techniques:


- Collaborative data modeling requires open-minded people. Data modelers must be prepared to meet regularly with stakeholders (take on a business analyst role) while business analysts and stakeholders must be willing to actively participate in some data modeling too. Everyone involved needs simple frameworks, checklists and guidelines that encourage ***interaction*** and prompt them through unfamiliar territory.

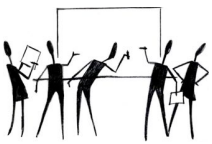
Collaborative modelers require techniques that encourage ***interaction***

- Business stakeholders have little appetite for traditional data models, even conceptual models (see **Data Model Types**, shortly) that are supposedly targeted at them. They find the ER diagrams and notation favored by data modelers (and generated by database modeling tools) too complex or too abstract. To engage stakeholders, agile modelers need to create less abstract, more ***inclusive*** data models using simple tools that are easy to use, and easy to share. These inclusive models must easily translate into the more technically detailed,

Collaborative data modeling must use simple, ***inclusive*** notation and tools

logical and physical, star schemas used by database administrators (DBAs) and ETL/BI developers.

Data modeling sessions (*model-storms*) need to be ***quick***: hours rather than days



- To encourage collaboration and support iteration, agile data modeling needs to be ***quick***. If stakeholders are going to participate in multiple modeling sessions they don't want each one to take days or weeks. Agile modelers want speed too. They don't want to wear out their welcome with stakeholders. The best results are obtained by modeling with groups of stakeholders who have the experience and knowledge to define common business terms (*conformed dimensions*) and prioritize requirements. It is hard enough to schedule long meetings with these people individually let alone in groups. Agile data modeling techniques must support *modelstorming*: impromptu stand up modeling that is quicker, simpler, easier and more fun than traditional approaches.

Agile modelers must balance ***JIT*** and ***JEDUF*** modeling to reduce design rework

- Stakeholders don't want to feel that a design is constantly iterating (fixing what they have already paid for) when they want to be incrementing (adding functionality). They want to see obvious progress and visible results. Agile modelers need techniques that ***support JIT*** modeling of current data requirement in details ***and JEDUF*** modeling of 'the big picture' to help anticipate future iterations and reduce the amount of design rework.

Evolutionary DW development benefits from ETL/BI tools that support automated testing

- Developers need to ***embrace database change***. They are used to working with (notionally) stable database designs, by-products of BDUF data modeling. It is support staff who are more familiar with coding around the database changes needed to match users' real requirements. To respond efficiently to evolutionary data warehouse design, agile ETL and BI developers need tools that support database impact analysis and automated testing.

DW designers must ***embrace change*** and allow their models to evolve

- Data warehouse designers also need to ***embrace data model change***. They will naturally want to limit the amount of disruptive database refactoring required by evolutionary design, but they must avoid resorting to generic data model patterns which reduce understandability and query performance, and can alienate stakeholders. Agile data warehouse modelers need dimensional design patterns that they can trust to represent tomorrow's BI requirements *tomorrow*, while they concentrate on today's BI requirements *now*.

Agile dimensional modeling techniques exist for addressing these requirements

If agile dimensional modeling that is ***interactive, inclusive, quick, supports JIT and JEDUF, and enables DW teams to embrace change*** seems like a tall order don't worry; while there are no silver bullets that will make everyone or everything agile overnight, there are proven tools and techniques that can address the majority of these agile modeling prerequisites.

# BEAM✲

BEAM✲ is an agile dimensional modeling method

BEAM✲ is an agile data modeling method for designing dimensional data warehouses and data marts. BEAM stands for Business Event Analysis & Modeling. As the name suggests it combines analysis techniques for gathering business event related data requirements and data modeling techniques for database design. The trailing ✲ (six point open centre asterisk) represents its dimensional deliverables: star schemas and the dimensional position of each of the *7Ws* it uses.

BEAM✲ is used to discover and document business event details

BEAM✲ consists of a set of repeatable, collaborative modeling techniques for rapidly *discovering* business event details and an inclusive modeling notation for *documenting* them in a tabular format that is easily understood by business stakeholders and readily translated into logical/physical dimensional models by IT developers.

## Data Stories and the *7Ws* Framework

BEAM✲ modelers and BI stakeholders use the *7Ws* to tell *data stories*

BEAM✲ gets BI stakeholders to think beyond their current reporting requirements by asking them to describe *data stories*: narratives that tease out the dimensional details of the business activity they need to measure. To do this BEAM✲ modelers ask questions using a simple framework based on the *7Ws*. By using the *7Ws* (*who, what, where, when, how many, why* and *how*) BEAM✲ conditions everyone involved to think dimensionally. The questions that BEAM✲ modelers ask stakeholders are the same types of questions that the stakeholders themselves will ask of the data warehouse when they become BI users. When they do, they will be thinking of *who, what, when, where, why* and *how* question combinations that measure their business.

## Diagrams and Notation

BEAM✲ tables support *data modeling by example*

*Example data tables* (or *BEAM✲ tables*) are the primary BEAM✲ modeling tool and diagram type. BEAM✲ tables are used to capture data stories in tabular form and describe data requirements using example data. By doing so they support collaborative *data modeling by example* rather than by abstraction. BEAM✲ tables are typically built up column by column on whiteboards from stakeholders' responses to the *7Ws* and are then documented permanently using spreadsheets. The resulting BEAM✲ models look more like tabular reports (see Figure 1-9) rather than traditional data models.

### BEAM✲ (Example Data) Tables

BEAM✲ tables look like simple tabular reports

BEAM✲ tables help engage stakeholders who would rather define reports that answer their specific business questions than do data modeling. While example data tables are not reports, they are similar enough for stakeholders to see them as

visible signs of progress. Stakeholders can easily imagine sorting and filtering the low-level detail columns of a business event using the higher-level dimensional attributes that they subsequently model.

**Figure 1-9**
Customer Orders BEAM✲ table

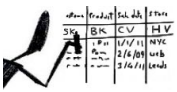
**CUSTOMER ORDERS [DE]**

| CUSTOMER | **orders** PRODUCT | on ORDER DATE | QUANTITY | for REVENUE | with DISCOUNT | using ORDER ID |
|---|---|---|---|---|---|---|
| [who] | [what] MD, GD | [when] MD | [Retail Units] | [$, £, €] | [$, £, €, %] | [how] GD |
| Elvis Priestley | iPip Blue Suede | 18-May-2011 | 1 | $249 | 0 | ORD1234 |
| Vespa Lynd | POMBook Air | 29-Jun-2011 | 1 | £1,400 | 10% | ORD007 |
| Elvis Priestley | iPip Blue Suede | 18-May-2011 | 1 | $249 | 0 | ORD4321 |
| Phillip Swallow | iPOM Pro | 14-Oct-2011 | 1 | £2,500 | £150 | ORD0001 |
| Walmart | iPip G1 | 10 Years Ago | 750 | $200,000 | $10,000 | ORD0012 |
| US Senate | iPOM + Printer | Yesterday | 100 | $150,000 | $20,000 | ORD5466 |
| US Senate | iPip Touch | Yesterday | 100 | $25,000 | $1,000 | ORD5466 |

## BEAM✲ Short Codes

BEAM✲ uses short codes to capture technical data properties

BEAM✲ tables are simple enough not to get in the way when modeling with stakeholders, but expressive enough to capture real-world data complexities and ultimately document the dimensional modeling design patterns used to address them. To do this BEAM✲ models use *short* (alphanumeric) *codes*: (mainly) 2 letter abbreviations of data properties that can be recorded in spreadsheet cells, rather than graphical notation that would require specialist modeling tools. By adding short codes, BEAM✲ tables can be used to:



- Document dimensional attribute properties including history rules
- Document fact properties including aggregation rules
- Record data-profiling results and map data sources to requirements
- Define physical dimensional models: fact and dimension tables
- Generate star schemas

BEAM✲ short codes act as dimensional modeling shorthand

BEAM✲ *short codes* act as dimensional modelers' shorthand for documenting generic data properties such as data type and nullability, and specific dimensional properties such as slowly changing dimensions and fact additivity. Short codes can be used to annotate any BEAM✲ diagram type for technical audiences but can easily be hidden or ignored when modeling with stakeholders who are disinterested in the more technical details. Short codes and other BEAM✲ notation conventions will be highlighted in the text in bold. Appendix B provides a reference list of short codes.

## Comparing BEAM✲ and Entity-Relationship Diagrams

We will use Pomegranate Corp. examples to illustrate BEAM✲

Throughout this book we will be illustrating BEAM✲ in action with worked examples featuring the fictional Pomegranate Corporation (POM). We begin now by comparing an ER diagram representation of Pomegranate's order processing data model (Figure 1-10) with an equivalent BEAM✲ table for the Customer Orders event (Figure 1-9).

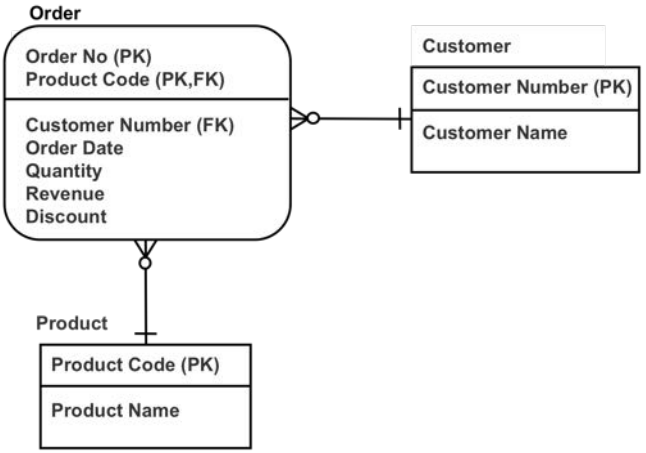



**Figure 1-10**
Order processing ER Diagram

By looking at the ERD you can tell that customers may place orders for multiple products at a time. The BEAM✲ table records the same information, but the example data also reveals the following:

Example data models capture more business information than ER models

- Customers can be individuals, companies, and government bodies.
- Products were sold yesterday.
- Products have been sold for 10 years.
- Products vary considerably in price.
- Products can be bundles (made up of 2 products).
- Customers can order the same product again on the same day.
- Orders are processed in both dollars and pounds.
- Orders can be for a single product or bulk quantities.
- Discounts are recorded as percentages and money.

Additionally, by scanning the BEAM✲ table you may have already guessed the type of products that Pomegranate sells and come to some conclusions as to what sort of company it is. Example data speaks volumes—wait until you hear what it says about some of Pomegranate's (fictional) staff!

Example data speaks volumes!

## Data Model Types

Agile dimensional modelers need to work with different types of models depending on the level of technical detail they are trying to capture or communicate and the technical bias of their collaborators and target audience. *Conceptual data models* (CDM) contain the least technical detail and are intended for exploring data requirements with non-technical stakeholders. *Logical data models* (LDM) allow modelers to record more technical details without going down to the database specific level, while *physical data models* (PDM) are used by DBAs to create database schemas for a specific DBMS. Table 1-2 shows the level of detail for each model type, its target audience on a DW/BI project, and the BEAM✲ diagram types that support that level of modeling.

*Conceptual*, *logical* and *physical* data models provide progressively more technical detail for more technical audiences

**Table 1-2**
Data Model Types

| DETAIL | CONCEPTUAL DATA MODEL | LOGICAL DATA MODEL | PHYSICAL DATA MODEL |
|---|---|---|---|
| **Entity Name** | ✓ | ✓ | |
| **Relationship** | ✓ | ✓ | |
| **Attribute** | Optional | ✓ | |
| **Cardinality** | Optional | ✓ | ✓ |
| **Primary Key** | | ✓ | ✓ |
| **Foreign Key** | | ✓ | ✓ |
| **Data Type** | | Optional | ✓ |
| **Table Name** | | | ✓ |
| **Column Name** | | | ✓ |
| **DW/BI Audience** | Data Modelers<br>Business Analysts<br>Business Experts<br>Stakeholders<br>BI Users | Data Modelers<br>ETL Developers<br>BI Developers | Data Modelers<br>DBAs<br>DBMS<br>ETL Developers<br>BI Developers |
| **BEAM✲ Diagram** | Example Data Table<br>Hierarchy Chart<br>Timeline<br>Event Matrix | Conceptual Diagrams with Short Codes<br>Enhanced Star Schema | Enhanced Star Schema<br>Event Matrix |

BEAM✲ and ER notation are jointly used to create collaborative models for different audiences

Based on the detail levels described in Table 1-2 the order processing ERD in Figure 1-10 is a logical data model as it shows primary keys, foreign keys and cardinality, while the BEAM✲ event in Figure 1-9 is a conceptual model (we prefer "business model") as this information is missing. With additional columns and short codes it could be added to the BEAM✲ table but each diagram type suits its target audience as is. BEAM✲ tables are more suitable for collaborative modeling with stakeholders than traditional ERD based conceptual models. While other BEAM✲ diagram types and short codes compliment and enhance ERDs for collaborating with developers on logical/physical star schema design.

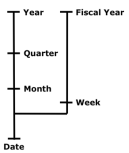
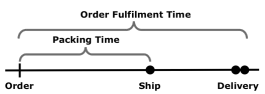
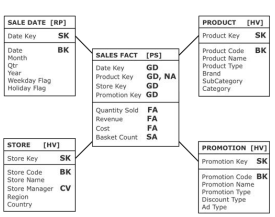
## BEAM✲ Diagram Types

BEAM✲ also uses event matrices, timelines, hierarchy charts and enhanced star schemas

Example data tables are not the only BEAM✲ modeling tools. BEAM✲ modelers also uses event matrices, hierarchy charts, timelines and enhanced star schemas to collaborate on various aspects of the design at different levels of business and technical detail. Table 1-3 summarizes the usage of each of the BEAM✲ diagram types, and lists their model types, audience and the chapter where they are described in detail.


**BEAM✲ supports the core agile values: *"Individuals and interactions over processes and tools."*, *"Working software over comprehensive documentation."* and *"Customer collaboration over contract negotiation."* BEAM✲ upholds these values and the agile principle of *"maximizing the amount of work not done"* by encouraging DW practitioners to work directly with stakeholders to produce compilable data models rather than requirements documents, and working BI prototypes of reports/dashboards rather than mockups.**

**Table 1-3** BEAM✲ Diagram Types

| DIAGRAM | USAGE | DATA MODEL TYPE | AUDIENCE | PRINCIPAL CHAPTER |
|---|---|---|---|---|
| **BEAM✲ (Example Data) Table**<br>CUSTOMER ORDERS [DE]<br>CUSTOMER \| orders PRODUCT \| on ORDER DATE \| for REVENUE<br>[who] \| [what] \| [when] \| [$, £, €]<br>Elvis Priestley \| iPip Blue Suede \| 18-May-2011 \| $249<br>Vespa Lynd \| POMBook Air \| 29-Jun-2011 \| £1,400<br>Elvis Priestley \| iPip Blue Suede \| 18-May-2011 \| $249<br>Phillip Swallow \| iPOM Pro \| 14-Oct-2011 \| £2,500 | Modeling business events and dimensions one at a time using example data to document their *7Ws* details.<br><br>Example data tables are also used to describe physical dimension and fact tables and explain dimensional design patterns. | Business<br>Logical<br>Physical | Data Modelers<br>Business Analysts<br>Business Experts<br>Stakeholders<br>BI Users | 2 |
| **Hierarchy Chart**<br>  | Discovering hierarchical relationships within dimensions and prompting stakeholders for dimensional attributes.<br><br>Hierarchy charts are also used to help define BI drill-down settings and aggregation levels for report and OLAP cube definition. | Business | Data Modelers<br>Business Analysts<br>Business Experts<br>Stakeholders<br>BI Users | 3 |
| **Timeline**<br>  | Exploring time relationships between business events.<br><br>Timelines are used to discover *when* details, process sequences and duration facts for measuring process efficiency. | Business | Data Modelers<br>Business Analysts<br>Business Experts<br>Stakeholders<br>BI Users | 8 |
| **Event Matrix**<br>  | Documents the relationships between all the events and dimensions within a model.<br><br>Event matrices record events in value-chain sequences and promote the definition and reuse of conformed dimensions across dimensional models. They are used instead of high-level ERDs to provide readable overviews of entire data warehouses or multi-star schema data marts. | Business<br>Logical<br>Physical | Data Modelers<br>Business Analysts<br>Business Experts<br>Stakeholders<br>BI Users<br>Data Modelers<br>ETL Developers<br>BI Developers | 4 |
| **Enhanced Star Schema**<br>  | Visualizing individual dimensional models and generating physical database schemas.<br><br>Enhanced star schemas are standard stars embellished with BEAM✲ short codes to record dimensional properties and design techniques not directly supported by generic data modeling tools. | Logical<br>Physical | Data Modelers<br>DBAs<br>DBMS<br>ETL Developers<br>BI Developers<br>Testers | 5 |

# Summary

- Data warehouses and operational systems are fundamentally different. They have radically different database requirements and should be modeled using very different techniques.

- Dimensional modeling is the appropriate technique for designing high-performance data warehouses because it produces simpler data models—star schemas—that are optimized for business process measurement, query performance and understandability.

- Star schemas record and describe the measureable events of business processes as fact tables and dimensions. These are not arbitrary denormalized data structures. Instead they represent the combination of the 7Ws (who, what, when, where, how many, why and how) that fully describe the details of each business event. In doing so, fact tables represents verbs, while the facts (measures) they contain and the dimensions they reference represent nouns.

- Dimensional modeling's process-orientation supports agile development by creating database designs that can be delivered in star schema/business process increments.

- Even with the right database design techniques there are numerous analysis challenges in gathering detailed data warehousing requirements in a timely manner.

- Both data-driven and reporting-driven analysis are problematic, increasingly so, with DW/BI development becoming more proactive and taking place in parallel with agile operational application development.

- Iterative, incremental and collaborative data modeling techniques are agile alternatives to the traditional BI data requirements gathering.

- BEAM✲ is an agile data modeling method for engaging BI stakeholders in the design of their own dimensional data warehouses.

- BEAM✲ data stories use the 7Ws framework to discover, describe and document business events dimensionally.

- BEAM✲ modelers encourage collaboration by using simple modeling tools such as whiteboards and spreadsheets to create inclusive data models.

- BEAM✲ models use example data tables and alphanumeric short codes rather than ER data abstractions and graphical notation to improve stakeholder communication. These models are readily translated into star schemas.

- BEAM✲ is an ideal tool for modelstorming a dimensional data warehouse design.


# 2

# MODELING BUSINESS EVENTS

Think like a wise man but communicate in the language of the people.
*— William Butler Yeats (1865–1939)*

*Business events* are the individual actions performed by people or organizations during the execution of business processes. When customers buy products or use services, brokers trade stocks, and suppliers deliver components, they leave behind a trail of business events within the operational databases of the organizations involved. These business events contain the atomic-level measurable details of the business processes that DW/BI systems are built to evaluate.

Business events are the measureable atomic details of business processes

BEAM✲ uses business events as incremental units of data discovery/data modeling. By prompting business stakeholders to tell their event *data stories,* BEAM✲ modelers rapidly gather the clear and concise BI data requirements they need to produce efficient dimensional designs.

BEAM✲ modelers discover BI data requirements by telling *data stories*

In this chapter we begin to describe the BEAM✲ collaborative approach to dimensional modeling, and provide a step-by-step guide to discovering a business event and documenting its data stories in a BEAM✲ table: a simple tabular format that is easily translated into a star schema. By following each step you will learn how to use the *7Ws* (*who, what, when, where, how many, why, and how)* to get stakeholders thinking dimensionally about their business processes, and describing the information that will become the dimensions and facts of *their* data warehouse—one that they themselves helped to design!

This chapter is a step-by-step guide to using BEAM✲ tables and the *7Ws* to describe event details

**Chapter 2 Topics *At a Glance***

- **Data stories and story types: discrete, recurring and evolving**
- **Discovering business events: asking "Who does what?"**
- **Documenting events: using BEAM✲ Tables**
- **Describing event details: using the *7Ws* and stories themes**
- **Modelstorming with whiteboards: practical collaborative data modeling**


# 3

# MODELING BUSINESS DIMENSIONS

I keep six honest serving-men (They taught me all I knew);
Their names are What and Why and When And How and Where and Who.

— *Rudyard Kipling,* The Elephant's Child

Business events need dimensions to fully describe them for reporting purposes

Business events and their numeric measurements are only part of the agile dimensional modeling story. On their own, BEAM✲ event tables are not sufficient to design a data warehouse or even a data mart, because they do not contain all the descriptive attributes required for reporting purposes. For complete BI flexibility, stakeholders need both the atomic-level event details modeled so far *and* higher-level descriptions that allow those details to be analyzed in practical ways. The data structures that provide these descriptive attributes are dimensions.

BEAM✲ modelers draw *hierarchy charts* and tell *change stories* to define dimensions

In addition to the *7Ws* and example data tables, BEAM✲ uses *hierarchy charts* and *change stories* to discover and define dimensional attributes. Hierarchy charts are used to explore the hierarchical relationships between attributes that support BI drill-down analysis, while change stories allow stakeholders to describe their business rules for handling *slowly changing dimensions*.

This chapter shows you how to model dimensions from event story details

In this chapter we describe how these BEAM✲ tools and techniques are used to model complete dimension definitions from individual event details. We will use the CUSTOMER and PRODUCT event story details from Chapter 2 for our example dimension modelstorming with stakeholders.

**Chapter 3 Topics**
***At a Glance***

- **Modeling the dimensions of a business event**
- **Using the *7Ws* and BEAM✲ tables to define dimensional attributes**
- **Drawing hierarchy charts to model dimensional hierarchies**
- **Telling change stories to describe dimensional history**


# 4

# MODELING BUSINESS PROCESSES

The only reason for time is so that everything doesn't happen at once

*— Albert Einstein*

Designing a data warehouse or data mart for business process measurement demands that you quickly move beyond modeling single business events. All but the simplest business processes are made up of multiple business events and BI stakeholders invariably want to do cross-process analysis. When you modelstorm these multi-event requirements you soon notice two crucial things:

BI Stakeholders need multiple events for process measurement

- **Stakeholders model events chronologically.** As you complete one event, stakeholders naturally think of related events that immediately follow or precede it. These event sequences represent business processes and value chains that need to be measured end-to-end.

Events sequences represent business processes and value chains

- **Stakeholders describe different events using many of the same *7Ws*.** When you define an event in terms of its *7Ws*, stakeholders start thinking of other events with the same details, especially events that share its subject or object. These shared details, known to dimensional modelers as *conformed dimensions*, are the basis for cross-process analysis.

Events share common dimensions that support cross-process analysis

In this chapter we describe how an *event matrix*, the single most powerful BEAM✲ artifact, is used to storyboard the data warehouse: rapidly model multiple events, identify significant business processes and conformed dimensions, and prioritize their development.

The *event matrix* is an agile tool for modeling multiple events

**Chapter 4 Topics**
***At a Glance***

- **The importance of conformed dimensions for agile DW design**
- **Modelstorming event sequences with an event matrix**
- **Prioritizing event and dimension development using Scrum**
- **Modeling event stories with conformed dimensions and examples**


# 5

# MODELING STAR SCHEMAS

We are all in the gutter, but some of us are looking at the stars.
*— Oscar Wilde*

In this chapter we describe the star schema design process for converting BEAM✲ models into flexible and efficient dimensional data warehouse models.

This chapter is a guide to:

The agile approach that we take begins with *test-first design*, by using *data profiling* techniques to verify the BEAM✲ model against the data available in source systems. This results in an *annotated model* which documents source data characteristics and issues. This is used for *model review* with stakeholders and development *sprint planning* with the DW/BI team.

Verifying BEAM✲ models against available data sources

Next, the revised BEAM✲ model is translated into a logical dimensional model by adding *surrogate keys*. The resulting facts and dimensions are documented by drawing *enhanced star schemas* using a combination of BEAM✲ and ER notation.

Converting BEAM✲ models into star schemas

Finally, the star schemas are used to generate physical data warehouse schemas which are validated by BI prototyping and documented by creating a physical dimensional matrix.

Validating DW designs by prototyping

**Chapter 5 Topics *At a Glance***

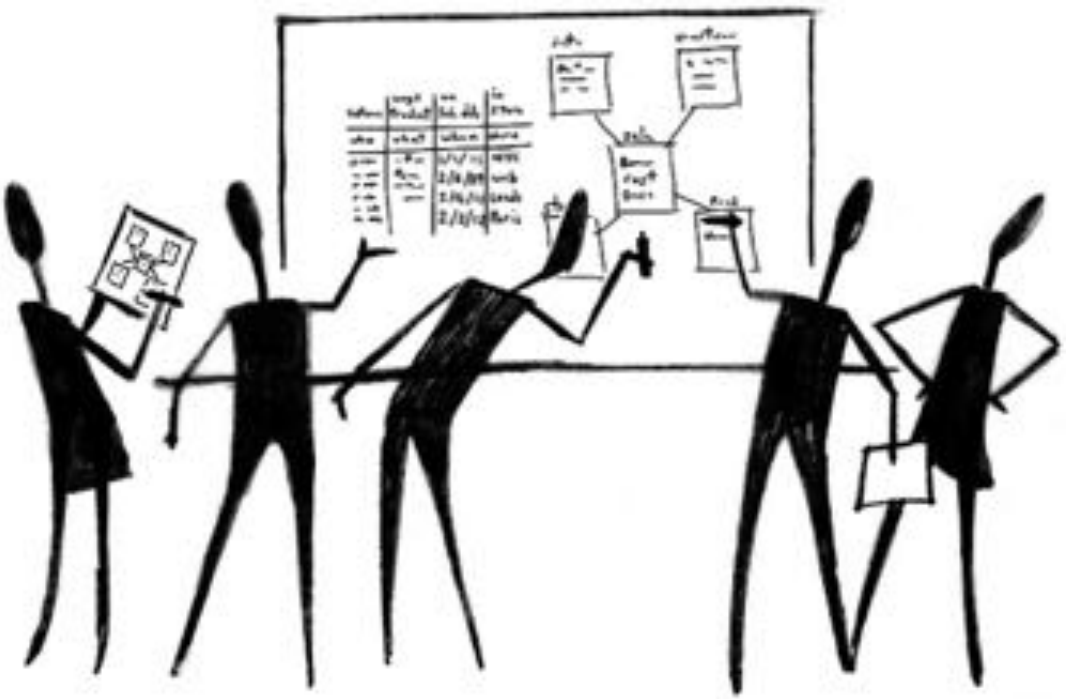
- **Data profiling to verify stakeholder data requirements**
- **Annotating BEAM✲ models with data sources and profile metrics**
- **Reviewing annotated models and planning development sprints**
- **Converting BEAM✲ models into logical/physical dimensional models**
- **The importance of data warehouse surrogate keys**
- **Designing for slowly changing dimensions**
- **Defining additive facts**
- **Drawing enhanced star schema diagrams and creating physical schemas**
- **BI Prototyping to validate dimensional models**
- **Creating a physical dimensional matrix**

# PART II: DIMENSIONAL DESIGN PATTERNS

**DIMENSIONAL MODELING TECHNIQUES FOR PERFORMANCE, FLEXIBILITY, AND USABILITY**

Computers are to design as microwaves are to cooking.

*— Milton Glaser*




# 6

# WHO AND WHAT

## Dimensional Design Patterns for People and Organizations, Products and Services

Who's on first?

*— Bud Abbott and Lou Costello*

What's next?

*— President Jed Bartlet, "The West Wing"*

*Who* and *what* are the most important dimensions

*Who* and *what* dimensions such as CUSTOMER, EMPLOYEE and PRODUCT represent some of the most interesting, highly scrutinized, and complex dimensions of a data warehouse. Modeling these dimensions and their inherent hierarchies presents a number of challenges that can be addressed by design patterns.

This chapter describes design patterns for defining flexible, high performance *who* and *what* dimensions

In the first of our *W*-themed design pattern chapters we begin by describing *mini-dimensions and snowflaking* for handling large, volatile customer dimensions, *swappable dimensions* for mixed customer type models and *hierarchy maps* for recursive customer relationships. We then move on to employee dimensions to cover *hybrid SCD views* for current value/historic value (**CV/HV**) reporting requirements and *multi-valued hierarchy maps* for multi-parent HR hierarchies with dotted-line relationships. We finish by looking at product and service dimension issues and introduce *multi-level dimensions* for variable detail facts and *reverse hierarchy maps* for component analysis.

**Chapter 6 Design Challenges *At a Glance***

- **Large, rapidly changing customer populations**
- **Mixed business models: businesses and consumers, products and services**
- **Simultaneous current and historic value reporting requirements**
- **Variable-depth hierarchies, recursive relationships**
- **Multi-valued hierarchies**
- **Business processes with variable levels of dimensional detail**
- **Product bill of materials**


# 7

# When and Where

## Dimensional Design Patterns for Time and Location

The past is a foreign country: they do things differently there.
*— L.P. Hartley, The Go-Between*

Every business event happens at a point in *time* or represents an interval of time. Time is the primary way that BI queries group ("show me monthly totals"), filter ("show me sales for Financial Q1"), and compare business events ("How are we doing year to date, versus last year?"). That is why *every* fact table has at least one time (*when*) dimension.

Time is the most frequently used dimension for BI analysis

Most business events occur at a specific geographical or online *location*. Many interesting events represent changes of location. Hence, a large number of fact tables have distinct *where* dimensions in addition to the location attributes that can be found in *who* and *what* dimensions, such as customer and product.

Location dimensions and attributes are frequently used too

Although *when* and *where* are separate dimensions, they can influence one another: Time zones, holidays and seasons, are all examples of *location-specific time attributes* that are affected by *event geography*. Similarly, analytically significant locations such as the first and last locations in a sequence of events are *timing-specific location dimensions*, affected by event chronology.

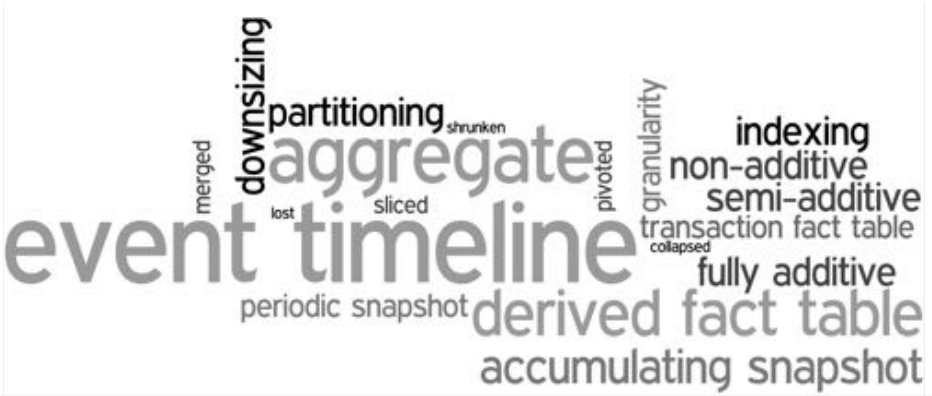
Time and location are separate dimensions but can affect one another

In this chapter, we describe dimensional design patterns for efficiently handling time and location, in particular, patterns for correctly analyzing year-to-date facts, and journeys—facts that represent changes in space and time, that are all about *where* and *when*.

This chapter describes *when* and *where* patterns

**Chapter 7 Design Challenges *At a Glance***

- **Efficient date and time reporting**
- **Correct year-to-date analysis**
- **Time zones, international holidays and seasons**
- **National language support**
- **Trip and journey analysis**




# 8

# How Many

## Design Patterns for High Performance Fact Tables and Flexible Measures

How many times must a man look up…
*— Bob Dylan, Blowin' in the Wind*

Everything that can be counted does not necessarily count;
everything that counts cannot necessarily be counted.
*— Albert Einstein*

In this chapter we describe how the three fact table patterns—t*ransaction fact tables, periodic snapshots, and accumulating snapshots*—are implemented to efficiently measure discrete, recurring and evolving business events. We particularly focus on the agile design of accumulating snapshots, by describing how the requirements for these powerful but complex fact tables can be visually modeled as evolving events using *event timelines*, our final BEAM✲ modelstorming tool. We also describe the BEAM✲ notation for capturing *fact additivity* and fully documenting the limitations of *semi-additive facts*, such as balances. We conclude with techniques for optimizing fact table performance and multi-fact table reporting by concentrating on design patterns for *aggregates* and other *derived fact tables* that accelerate and simplify BI queries

This chapter covers techniques for incrementally designing and developing high-performance fact tables and flexible measures

**Chapter 8 Design Challenges *At a Glance***

- **Point in time event measurement**
- **Periodic measurement**
- **Evolving process measurement**
- **Modeling evolving event milestones and duration measures**
- **Incremental development of complex fact tables**
- **Flexible fact definition**
- **Fact table performance**
- **Correctly querying multiple fact tables at once**
- **Cross-process analysis using simple BI tools**


# 9

# WHY AND HOW

## Dimensional Design Patterns for Cause and Effect

There is occasions and causes why and wherefore in all things.
*— William Shakespeare (1564–1616), "King Henry V", Act 5, scene 1*

How am I doing?
*— Ed Koch, Mayor of New York 1978–1989*

*Why* and *how* dimensions are closely linked: they describe *cause* and *effect*

Some of the most valuable dimensions in a data warehouse attempt to explain why and how events occur. *Why* dimensions are used to describe direct and indirect causal factors. They are often closely linked to the *how* dimensions that provide all the remaining event descriptions that are not related to the major *who, what, when* and *where* dimension types. Together *why* and *how* represent *cause* and *effect* and complete the *7W* dimensional description of a business event.

This chapter describes *why* and *how* dimension design patterns

In our final chapter we cover dimensional design patterns for describing *how* events occur and *why* facts vary. We focus particularly on bridge table patterns for representing multiple causal factors and multi-valued dimensions in general. We describe how bridge table weighting factors are used to preserve atomic fact granularity and avoid ETL time fact allocations. We also describe how bridge tables can be augmented with multi-level dimensions and pivoted dimensions to efficiently handle *barely* multi-valued reporting and complex combination constraints. We conclude with step, range band and audit dimension techniques for analyzing sequential events, grouping by facts and handling ETL metadata.

**Chapter 9 Design Challenges *At a Glance***

- **Direct and indirect causal factors**
- **Attributing multiple causes to a fact**
- **Dealing with *barely* multi-valued dimensions efficiently**
- **Handling complex combination constraints**
- **Understanding sequential behavior**
- **Range band reporting**
- **Tracking data quality and lineage**

# Websites

**decisionone.co.uk** : DecisionOne Consulting, Lawrence Corr's training and consulting firm.

**llumino.com** : Llumino, Jim Stagnitto's consulting firm.

**modelstorming.com** : The companion website to this book where you can download the BEAM✲ *Modelstormer* spreadsheet, the *BI Model Canvas* (inspired by the *Business Model Canvas*) plus other useful BEAM✲ tools and example models from the book and beyond. It also contains links to our recommended books, articles, websites, and training courses.