**Dimensions:**

1. You should make standard decodes  
   for the operational codes available as dimension attributes to provide consistent  
   labeling on queries, reports, and BI applications. The decode values should never be  
   buried in the reporting applications where inconsistency is inevitable.
2. In many ways, the data warehouse is only as good as the dimension attributes; the  
   analytic power of the DW/BI environment is directly proportional to the quality and  
   depth of the dimension attributes.
3. Dimensions provide the entry points to the data, and the final labels and  
   groupings on all DW/BI analyses.
4. The designer’s dilemma of whether a numeric quantity is a fact or a  
   dimension attribute is rarely a difficult decision. Continuously valued numericobservations are almost always facts; discrete numeric observations drawn from a  
   small list are almost always dimension attributes.
5. You should resist the perhaps habitual  
   urge to normalize data by storing only the brand code in the product dimension and  
   creating a separate brand lookup table, and likewise for the category description in a  
   separate category lookup table. This normalization is called *snowflaking*. Instead of  
   third normal form, dimension tables typically are highly denormalized with flattened  
   many-to-one relationships within a single dimension table. Because dimension tables  
   typically are geometrically smaller than fact tables, improving storage efficiency by  
   normalizing or snowflaking has virtually no impact on the overall database size. You  
   should almost always trade off dimension table space for simplicity and accessibility.

Table

Description automatically generated

**Star Schema:**

Diagram, table

Description automatically generated

1. The simplicity of a dimensional model also has performance benefits. Database  
   optimizers process these simple schemas with fewer joins more efficiently.
2. A database engine can make strong assumptions about first constraining the heavily  
   indexed dimension tables, and then attacking the fact table all at once with the  
   Cartesian product of the dimension table keys satisfying the user’s constraints.  
   Amazingly, using this approach, the optimizer can evaluate arbitrary n-way joins  
   to a fact table in a single pass through the fact table’s index.

Ex:

* 1. (a,b) |><| (b,c) |><| (d,e) is a 3-way join
  2. (a,b) |><| (b,c) |><| (d,e) |><| (e,f) |><| (f,g) is a 5-way join

1. Every dimension is equivalent; all dimensions are symmetrically equal entry points into the fact table.
2. This book illustrates repeatedly that the most granular or atomic data has the  
   most dimensionality. Atomic data that has not been aggregated is the most expressive data; this atomic data should be the foundation for every fact table design to  
   withstand business users’ ad hoc attacks in which they pose unexpected queries.

Diagram

Description automatically generated

1. You can easily envision the SQL that’s written (or more likely generated by a BI  
   tool) to create this report:  
   Text

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

**Kimball DW/BI Architecture:**

Diagram, schematic

Description automatically generated

1. Unfortunately, some DW/BI initiatives have failed miserably because they focused  
   all their energy and resources on constructing the normalized structures rather  
   than allocating time to developing a dimensional presentation area that supports  
   improved business decision making. Although enterprise-wide data consistency is a  
   fundamental goal of the DW/BI environment, there may be effective and less costly  
   approaches than physically creating normalized tables in the ETL system, if these  
   structures don’t already exist.
2. It is acceptable to create a normalized database to support the ETL  
   processes; however, this is not the end goal. The normalized structures must be  
   off-limits to user queries because they defeat the twin goals of understandability  
   and performance.
3. We have several strong opinions about the presentation area. First of all, we insist  
   that the data be presented, stored, and accessed in dimensional schemas, either  
   relational star schemas or OLAP cubes. Fortunately, the industry has matured to the  
   point where we’re no longer debating this approach; it has concluded that dimensional modeling is the most viable technique for delivering data to DW/BI users.
4. Although the presentation area also may contain  
   performance-enhancing aggregated data, it is not sufficient to deliver these summaries without the underlying granular data in a dimensional form. In other words,  
   it is completely unacceptable to store only summary data in dimensional models  
   while the atomic data is locked up in normalized models. It is impractical to expect  
   a user to drill down through dimensional data almost to the most granular level and  
   then lose the benefits of a dimensional presentation at the final step. Although DW/  
   BI users and applications may look infrequently at a single line item on an order,  
   they may be very interested in last week’s orders for products of a given size (or  
   flavor, package type, or manufacturer) for customers who first purchased within the last 6 months (or reside in a given state or have certain credit terms). The most  
   finely grained data must be available in the presentation area so that users can ask  
   the most precise questions possible. Because users’ requirements are unpredictable  
   and constantly changing, you must provide access to the exquisite details so they  
   can roll up to address the questions of the moment.

**Presentation area to support BI**

1. Dimensional models should correspond to physical data capture  
   events; they should not be designed to deliver the report-of-the-day. An enterprise’s  
   business processes cross the boundaries of organizational departments and functions. In other words, you should construct a single fact table for atomic sales metrics  
   rather than populating separate similar, but slightly different, databases containing  
   sales metrics for the sales, marketing, logistics, and finance teams.
2. Dimensional models should correspond to physical data capture  
   events; they should not be designed to deliver the report-of-the-day.
3. All the dimensional structures must be built using common, conformed dimensions. This is the basis of the *enterprise data warehouse bus architecture.*
4. Adherence to the bus architecture is the final stake in the ground  
   for the presentation area. Without shared, conformed dimensions, a dimensional  
   model becomes a standalone application. Isolated stovepipe data sets that cannot be  
   tied together are the bane of the DW/BI movement as they perpetuate incompatible  
   views of the enterprise.
5. If you have any hope of building a robust and integrated  
   DW/BI environment, you must commit to the enterprise bus architecture. When  
   dimensional models have been designed with conformed dimensions, they can be  
   readily combined and used together.
6. The presentation area in a large enterprise  
   DW/BI solution ultimately consists of dozens of dimensional models with many of  
   the associated dimension tables shared across fact tables.
7. Using the bus architecture is the secret to building distributed DW/BI systems.  
   When the bus architecture is used as a framework, you can develop the enterprise  
   data warehouse in an agile, decentralized, realistically scoped, iterative manner.
8. Data in the queryable presentation area of the DW/BI system must be  
   dimensional, atomic (complemented by performance-enhancing aggregates), business process-centric, and adhere to the enterprise data warehouse bus architecture.  
   The data must not be structured according to individual departments’ interpretation of the data.Diagram

   Description automatically generated

**Alternative DI/DW Architectures:**

1. **Independent Data Mart Architecture**  
   With this approach, analytic data is deployed on a departmental basis without concern to sharing and integrating information across the enterprise  
   Diagram

   Description automatically generated
   1. When business users from these two departments discuss organizational performance based on reports from their respective repositories, not surprisingly, none of  
      the numbers match because of the differences in business rules and labelling.
   2. Although no industry leaders advocate these independent  
      data marts, this approach is prevalent, especially in large organizations. It mirrors  
      the way many organizations fund IT projects, plus it requires zero cross-organizational data governance and coordination. It’s the path of least resistance for fast  
      development at relatively low cost, at least in the short run.
   3. So our concepts of dimensional modeling are often applied  
      in this architecture, despite the complete disregard for some of our core tenets, such  
      as focusing on atomic details, building by business process instead of department,  
      and leveraging conformed dimensions for enterprise consistency and integration.
2. **Hub and Spoke Corporate Information Factory   
   Inmon Architecture**

Diagram

Description automatically generated

* 1. With the CIF, data is extracted from the operational source systems and processed  
     through an ETL system sometimes referred to as data acquisition. The atomic data  
     that results from this processing lands in a 3NF database; this normalized, atomic  
     repository is referred to as the Enterprise Data Warehouse (EDW) within the CIF  
     architecture. Although the Kimball architecture enables optional normalization to  
     support ETL processing, the normalized EDW is a mandatory construct in the CIF.  
     Like the Kimball approach, the CIF advocates enterprise data coordination and integration. The CIF says the normalized EDW fills this role, whereas the Kimball architecture stresses the importance of an enterprise bus with conformed dimensions.
  2. Organizations who have adopted the CIF approach often have business users  
     accessing the EDW repository due to its level of detail or data availability timeliness. However, subsequent ETL data delivery processes also populate downstream  
     reporting and analytic environments to support business users. Although often  
     dimensionally structured, the resultant analytic databases typically differ from  
     structures in the Kimball architecture’s presentation area in that they’re frequently  
     departmentally-centric (rather than organized around business processes) and populated with aggregated data (rather than atomic details). If the data delivery ETL  
     processes apply business rules beyond basic summarization, such as departmental  
     renaming of columns or alternative calculations, it may be difficult to tie these  
     analytic databases to the EDW’s atomic repository.

1. **Hybrid Hub-and-Spoke and Kimball Architecture**Diagram, schematic

   Description automatically generated
   1. this architecture populates a CIF-centric EDW that is completely off-limits to business users for analysis and reporting. It’s merely the source to populate a Kimball-esque presentation area  
      in which the data is dimensional, atomic (complemented by aggregates), process centric, and conforms to the enterprise data warehouse bus architecture.
   2. Some proponents of this blended approach claim it’s the best of both worlds. Yes, it  
      blends the two enterprise-oriented approaches. It may leverage a pre existing investment in an integrated repository, while addressing the performance and usability  
      issues associated with the 3NF EDW by offloading queries to the dimensional presentation area.
   3. If you’ve already invested in the creation of a 3NF EDW, but it’s not delivering  
      on the users’ expectations of fast and flexible reporting and analysis, this hybrid  
      approach might be appropriate for your organization.
   4. If you’re starting with a blank  
      sheet of paper, the hybrid approach will likely cost more time and money, both during development and ongoing operation, given the multiple movements of data and redundant storage of atomic details
   5. If you have the appetite, the perceived need, and  
      perhaps most important, the budget and organizational patience to fully normalize  
      and instantiate your data before loading it into dimensional structures that are well  
      designed according to the Kimball methods, go for it.

**Dimensional Modeling Myths:**

1. Summary data should complement the granular detail solely to provide improved performance for common queries, but not replace the details
2. Nothing about a dimensional model  
   prohibits storing substantial history. The amount of history available in dimensional  
   models must only be driven by the business’s requirements.
3. Rather than drawing boundaries based on organizational departments, dimensional  
   models should be organized around business processes, such as orders, invoices, and  
   service calls.
4. Multiple business functions often want to analyze the same metrics  
   resulting from a single business process. Multiple extracts of the same source data  
   that create multiple, inconsistent analytic databases should be avoided
5. The correct starting point for  
   your dimensional models is to express data at the lowest detail possible for maximum flexibility and extensibility.
6. Remember, when you pre-suppose the business question, you’ll likely pre-summarize the data, which can be fatal in the long run.
7. Dimensional models most certainly can be integrated if they conform to the enterprise  
   data warehouse bus architecture.
8. Conformed dimensions are built and maintained  
   as centralized, persistent master data in the ETL system and then reused across  
   dimensional models to enable data integration and ensure semantic consistency.

**Think Dimensionally:**

1. When specifying the project’s scope, you must stand firm to focus on a single business process per project and not sign up to deploy a dashboard that covers a handful of them in a single iteration.
2. When prioritizing opportunities and developing the DW/BI roadmap, business processes are the unit of work.

**Modeling Techniques – Fundamental Concepts:**

1. Gather business requirements and Data Realities
   1. uncover the requirements via sessions with business representatives to understand  
      their objectives based on key performance indicators, compelling business issues,  
      decision-making processes, and supporting analytic needs.
   2. At the same time, data  
      realities are uncovered by meeting with source system experts and doing high-level  
      data profiling to assess data feasibilities.
2. Collaborative data modelling workshops
3. Four Key Decisions during the design of a dimensional model:
   1. Select the business process
      1. *Business processes* are the operational activities performed by your organization,  
         such as taking an order, processing an insurance claim, registering students for a  
         class, or snapshotting every account each month.
      2. Business process events generate or capture performance metrics that translate into facts in a fact table.
      3. Each business process corresponds to a row in the enterprise data warehouse bus matrix.
   2. Declare the grain
      1. The grain must be declared before choosing dimensions or facts because every candidate dimension or fact must be consistent with the grain.
      2. *Atomic grain* refers to the lowest level at which data is captured by a given business process.
      3. rolled-up summary grains are important for performance tuning, but they  
         pre-suppose the business’s common questions.
      4. Each proposed fact table grain results in a separate physical table; different grains must not be mixed in the same fact table.
   3. Identify the dimensions
   4. Identify the facts
   5. Business Rules

**Star Schema and OLAP Cubes:**

1. An *online analytical processing (OLAP) cube* is a dimensional structure implemented in a multidimensional database; it can be equivalent in content to, or more often derived from, a relational star schema.  
   An OLAP cube contains dimensional attributes and facts, but it is accessed through languages with more analytic capabilities than SQL, such as XMLA and MDX. OLAP

**Additive, Semi Additive and Non-additive Facts**

1. The numeric measures in a fact table fall into three categories.
2. additive measures can be summed across any of the dimensions associated with the fact table.
3. *Semi-additive* measures can be summed across some dimensions, but not all.
4. balance amounts are common semi-additive facts because they are additive across all dimensions except time.
5. Finally, some measures are completely *non-additive*, such as ratios.

**Null in Fact Tables**

1. Null-valued measurements behave gracefully in fact tables. The aggregate functions  
   (SUM, COUNT, MIN, MAX, and AVG) all do the “right thing” with null facts. However,  
   nulls must be avoided in the fact table’s foreign keys because these nulls would  
   automatically cause a referential integrity violation. Rather than a null foreign key,  
   the associated dimension table must have a default row (and surrogate key) representing the unknown or not applicable condition.

**Transaction Fact Tables**

1. A row in a *transaction fact table* corresponds to a measurement event at a point in  
   space and time. Atomic transaction grain fact tables are the most dimensional and  
   expressive fact tables; this robust dimensionality enables the maximum slicing  
   and dicing of transaction data.

**Periodic Snapshot Fact Tables**

1. A row in a periodic snapshot fact table summarizes many measurement events occurring over a standard period, such as a day, a week, or a month. The grain is the  
   period, not the individual transaction. Periodic snapshot fact tables often contain  
   many facts because any measurement event consistent with the fact table grain is  
   permissible. These fact tables are uniformly dense in their foreign keys because  
   even if no activity takes place during the period, a row is typically inserted in the  
   fact table containing a zero or null for each fact.

**Accumulating Snapshot Fact Tables**

1. A row in an *accumulating snapshot fact table* summarizes the measurement events  
   occurring at predictable steps between the beginning and the end of a process.
2. Pipeline or workflow processes, such as order fulfillment or claim processing, that  
   have a defined start point, standard intermediate steps, and defined end point can be  
   modeled with this type of fact table.
3. There is a date foreign key in the fact table for each critical milestone in the process.
4. An individual row in an accumulating snapshot fact table, corresponding for instance to a line on an order, is initially inserted when the order line is created. As pipeline progress occurs, the accumulating fact  
   table row is revisited and updated.
5. This consistent updating of accumulating snapshot fact rows is unique among the three types of fact tables.
6. They often include numeric lag measurements consistent with the grain, along with milestone completion counters.

**Factless Fact Tables**

1. Although most measurement events capture numerical results, it is possible that  
   the event merely records a set of dimensional entities coming together at a moment  
   in time.
2. For example, an event of a student attending a class on a given day may  
   not have a recorded numeric fact, but a fact row with foreign keys for calendar day,  
   student, teacher, location, and class is well-defined.
3. Likewise, customer communications are events, but there may be no associated metrics.

**Aggregate Fact Tables or OLAP Cubes**

1. These aggregate fact tables should be available to  
   the BI layer at the same time as the atomic fact tables so that BI tools smoothly  
   choose the appropriate aggregate level at query time.
2. This process, known as  
   *aggregate navigation*, must be *open* so that every report writer, query tool, and BI  
   application harvests the same performance benefits.
3. Aggregate fact tables contain foreign keys to shrunken conformed dimensions, as  
   well as aggregated facts created by summing measures from more atomic fact tables.

**Consolidated Fact Tables**

1. It is often convenient to combine facts from multiple processes together into a single  
   *consolidated fact table* if they can be expressed at the same grain.
2. For example, sales actuals can be consolidated with sales forecasts in a single fact table to make the task  
   of analyzing actuals versus forecasts simple and fast, as compared to assembling a  
   drill-across application using separate fact tables.
3. Consolidated fact tables add burden to the ETL processing, but ease the analytic burden on the BI applications. They should be considered for cross-process metrics that are frequently analyzed together.

**Dimension Table Structure**

1. Every dimension table has a single primary key column. This primary key is embedded  
   as a foreign key in any associated fact table where the dimension row’s descriptive  
   context is exactly correct for that fact table row.
2. Dimension table attributes are the primary target of constraints and grouping specifications from queries and BI applications.

**Drilling Down**

1. Drilling down is the most fundamental way data is analyzed by business users. Drilling  
   down simply means adding a row header to an existing query; the new row header  
   is a dimension attribute appended to the GROUP BY expression in an SQL query. The  
   attribute can come from any dimension attached to the fact table in the query. Drilling  
   down does not require the definition of predetermined hierarchies or drill-down paths.  
   See the section “Drilling Across.”

**Degenerate Dimensions**

1. Sometimes a dimension is defined that has no content except for its primary key.  
   For example, when an invoice has multiple line items, the line item fact rows inherit  
   all the descriptive dimension foreign keys of the invoice, and the invoice is left with  
   no unique content. But the invoice number remains a valid dimension key for fact  
   tables at the line item level.
2. This *degenerate dimension* is placed in the fact table with the explicit acknowledgment that there is no associated dimension table.
3. Degenerate dimensions are most common with transaction and accumulating snapshot fact tables.