07

Multidimensional Modeling - Order Management

Notice

- **Author**
  - **João Moura Pires (jmp@di.fct.unl.pt)**

-

-

Bibliography

- **Many examples are extracted and adapted from**
  - **The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition) - Ralph Kimball, Margy Ross**

Table of Contents

Introduction

Order Management

- Order management consists of several critical business processes:
  - Order,
  - Shipment,
  - Invoice processing.
- Business metrics:
  - Sales volume
  - Invoice revenue

Goal: to discuss some techniques

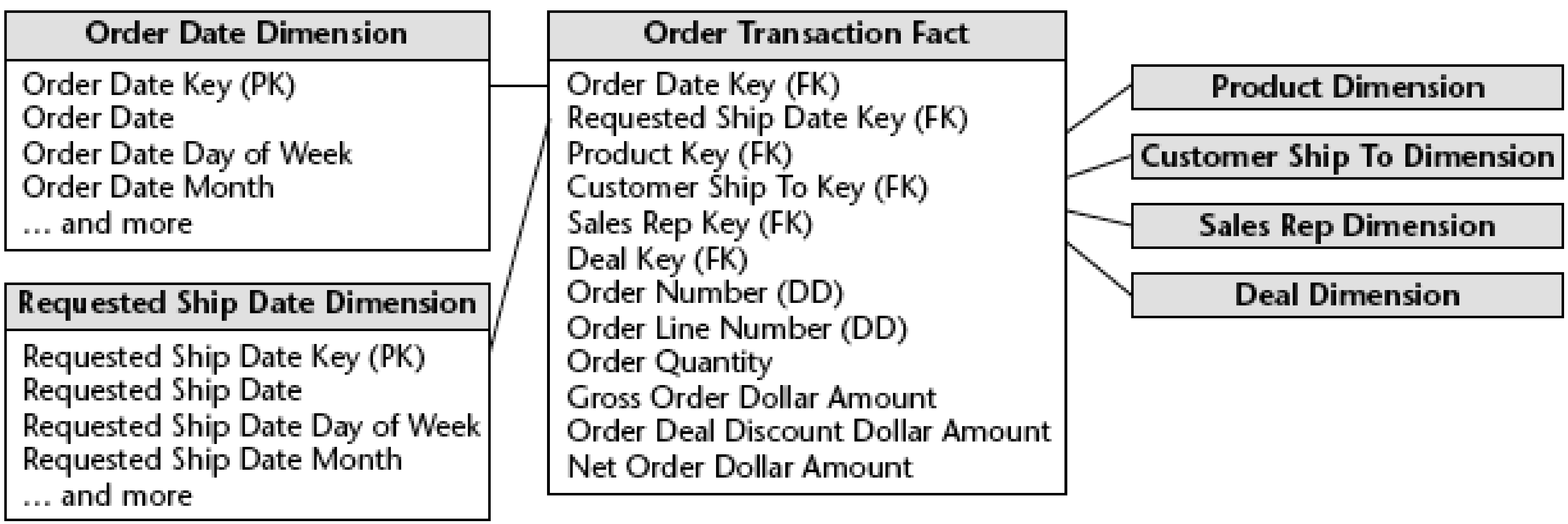
	Date	Product	Customer	Deal	Sales Rep	Ship From	Shipper
Quotes	X	X	X	X	X		
Orders	X	X	X	X	X		
Shipments	X	X	X	X	X	X	X
Invoicing	X	X	X	X	X	X	X

[Kimball, 2002]

Order Transactions

Order Transactions

- The **natural granularity** for an order transaction fact table:
  - One row for each line item on an order.
  - The facts associated with this process typically include the **order quantity**, **extended gross order dollar amount**, **order discount dollar amount**, and **extended net order dollar amount** (which is equal to the gross order amount less the discounts).



[Kimball, 2002]

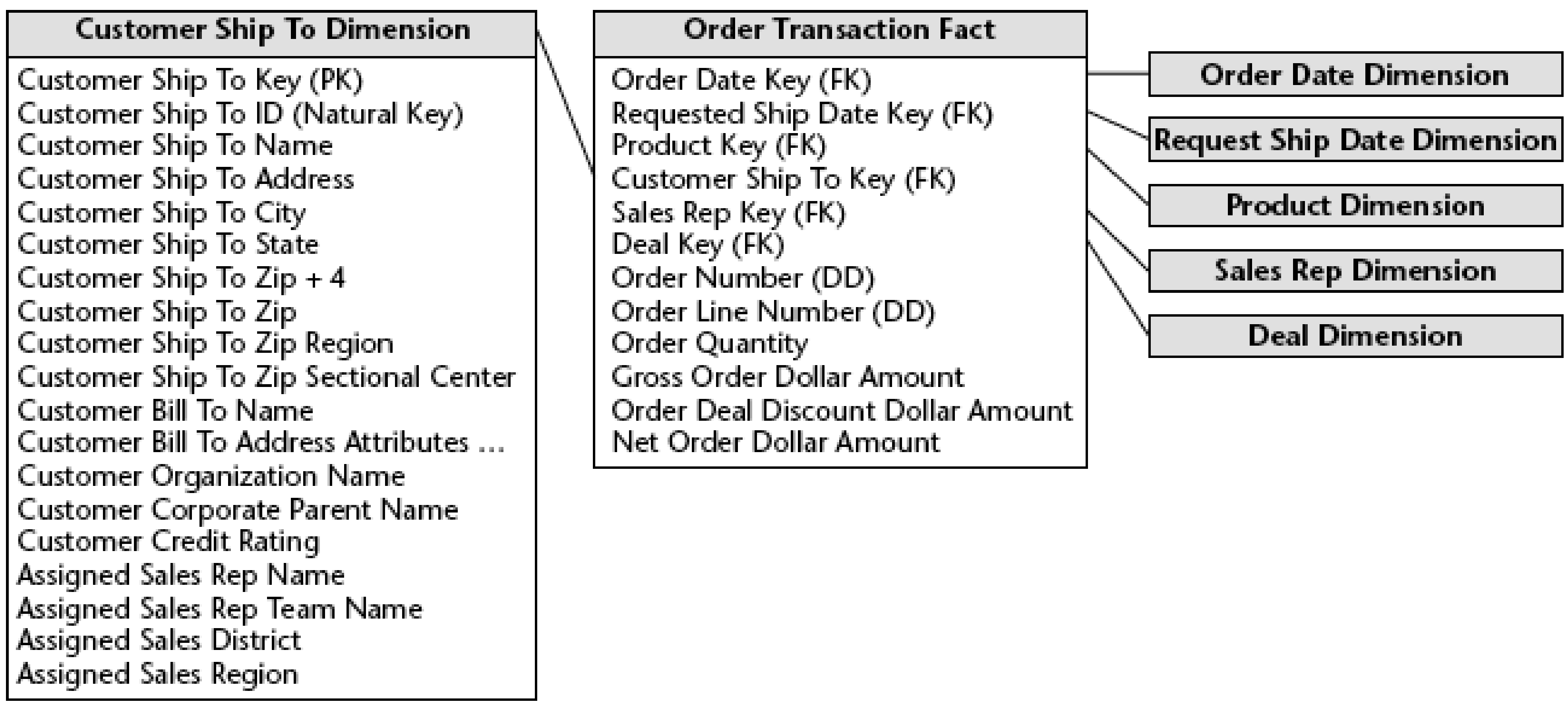
Fact Normalization

- To normalize the fact table so that there's a single, **generic fact amount**, along with a **dimension that identifies the type of fact**?
- This technique **may make sense** when:
  - The set of **facts** is **sparsely populated** for a given fact row
    - If we were to normalize the facts, we'd be **multiplying the number of rows** in the fact table **by the number of fact types**.
  - And **no computations** are made between facts.
    - if we are **performing any arithmetic function between the facts** (such as discount amount as a percentage of gross order amount), it is far easier if the facts are in the same row.

[Kimball, 2002]

Customer Ship-To Dimension

- The customer ship-to dimension contains **one row for each discrete location to which we ship a product**. Customer ship-to dimension tables can range from moderately sized (thousands of rows) to extremely large (millions of rows) depending on the nature of the business.



[Kimball, 2002]

Customer Ship-To Dimension

- **Several separate and independent hierarchies typically coexist** in a customer ship-to dimension:
  - Since the ship-to location is a point in space, any number of geographic hierarchies may be defined by nesting ever-larger geographic entities around the point.
  - Another common hierarchy is the **customer's organizational hierarchy**, assuming that the customer is a corporate entity. **For each customer ship-to, we might have a customer bill-to and customer corporation**.

> **It is natural and common, especially for customer-oriented dimensions, for a dimension to simultaneously support multiple independent hierarchies. The hierarchies may have different numbers of levels. Drilling up and drilling down within each of these hierarchies must be supported in a data warehouse.**

[Kimball, 2002]

Customer Ship-To Dimension

- The implied **assumption that multiple ship-tos roll up to a single bill-to in a many-to-one relationship may be wrong**. There are always a few exceptions involving **ship-tos that are associated with more than one bill-to**.
  - If this is a **rare occurrence**, it would be reasonable to generalize the customer ship-to dimension so that the grain of the dimension is each unique ship-to/bill-to combination
    - If there are two sets of bill-to information associated with a given ship-to location, then there would be two rows in the dimension, one for each combination.
  - If **many of the ship-tos are associated with many bill-tos** in a robust many-to-many relationship, then ship-to and bill-to probably need to be **handled as separate dimensions** that are linked together by the fact table.

[Kimball, 2002]

Customer - Sales organization

- Designers sometimes question whether **sales organization attributes** should be modeled as a **separate dimension** or the attributes **just should be added to the existing customer dimension.**
  - If sales reps are highly correlated with customer ship-tos in a one-to-one or many-to-one relationship, combining the sales organization attributes with the customer ship-to dimension is a viable approach.
  - However, sometimes the relationship between sales organization and customer ship-to is more complicated.

[Kimball, 2002]

Customer - Sales organization

- The **one-to-one or many-to-one relationship** may turn out to be a **many-to many** relationship. (see previous slide)
- If the **relationship** between **sales** rep and **customer** ship-to **varies over time** or under the **influence of a fourth dimension such as product**, then the combined dimension is in reality some kind of fact table itself!
  - In this case, create separate dimensions for the sales rep and the customer ship-to.
- If the sales rep and customer ship-to dimensions **participate independently** in **other business process fact tables**, we'd likely keep the dimensions separate.
  - Creating a single customer ship-to dimension with sales rep attributes exclusively around orders data may make some of the other processes and relationships difficult to express

[Kimball, 2002]

Customer - Sales organization

- When entities have a **fixed**, **time-invariant**, **strongly correlated relationship**, they obviously should be modeled as a single dimension.
- In **most other cases**, your design likely will be simpler and more manageable when you separate the entities into **two dimensions** (while remembering the general guidelines concerning too many dimensions).
- Users sometimes want the ability to analyze the complex assignment of sales reps to customers over time, even if no order activity has occurred. In this case, we could construct a **fact-less fact table**, to capture the **sales rep coverage,** even if some of the assignments never resulted in a sale.

[Kimball, 2002]

Junk Dimensions

- We are often left with a number of **miscellaneous indicators and flags**, each of which takes on a small range of discrete values.
  - Leave the flags and indicators unchanged in the fact table row.
    - This could cause the fact table row to swell alarmingly.
  - Make each flag and indicator into its own separate dimension.
    - Doing so could cause our 5-dimension design to balloon into a 25-dimension design.
  - Strip out all the flags and indicators from the design.
  - An **appropriate approach** for tackling these flags and indicators is to study them carefully and then **pack them into one or more junk dimensions**.

[Kimball, 2002]

Junk Dimensions

A junk dimension is a convenient grouping of typically low-cardinality flags and indicators. By creating an abstract dimension, we remove the flags from the fact table while placing them into a useful dimensional framework.

Order Indicator Key	Payment Type Description	Payment Type Group	Inbound/ Outbound Order Indicator	Commission Credit Indicator	Order Type Indicator
1	Cash	Cash	Inbound	Commissionable	Regular
2	Cash	Cash	Inbound	Non-Commissionable	Display
3	Cash	Cash	Inbound	Non-Commissionable	Demonstration
4	Cash	Cash	Outbound	Commissionable	Regular
5	Cash	Cash	Outbound	Non-Commissionable	Display
6	Discover Card	Credit	Inbound	Commissionable	Regular
7	Discover Card	Credit	Inbound	Non-Commissionable	Display
8	Discover Card	Credit	Inbound	Non-Commissionable	Demonstration
9	Discover Card	Credit	Outbound	Commissionable	Regular
10	Discover Card	Credit	Outbound	Non-Commissionable	Display
11	MasterCard	Credit	Inbound	Commissionable	Regular
12	MasterCard	Credit	Inbound	Non-Commissionable	Display
13	MasterCard	Credit	Inbound	Non-Commissionable	Demonstration
14	MasterCard	Credit	Outbound	Commissionable	Regular

[Kimball, 2002]

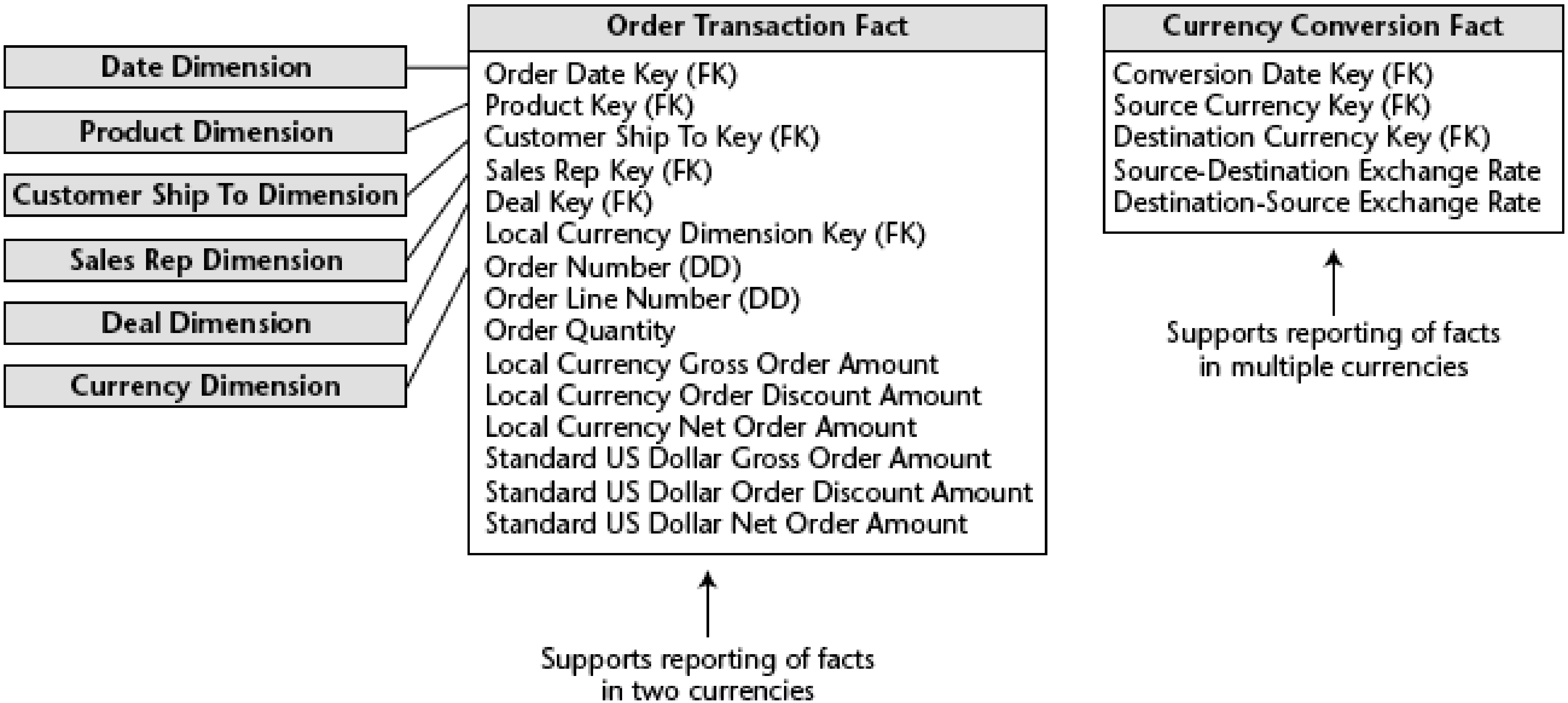
Multiple Currencies

- We may be capturing order transactions in more than 15 different currencies. We certainly wouldn't want to include columns in the fact table for each currency because theoretically there are an open-ended number of currencies.
- Requirements:
  - The order transactions be expressed in both local currency and the standardized corporate currency.
  - We may need to allow a manager in any country to see order volume in any currency.

[Kimball, 2002]

Multiple Currencies

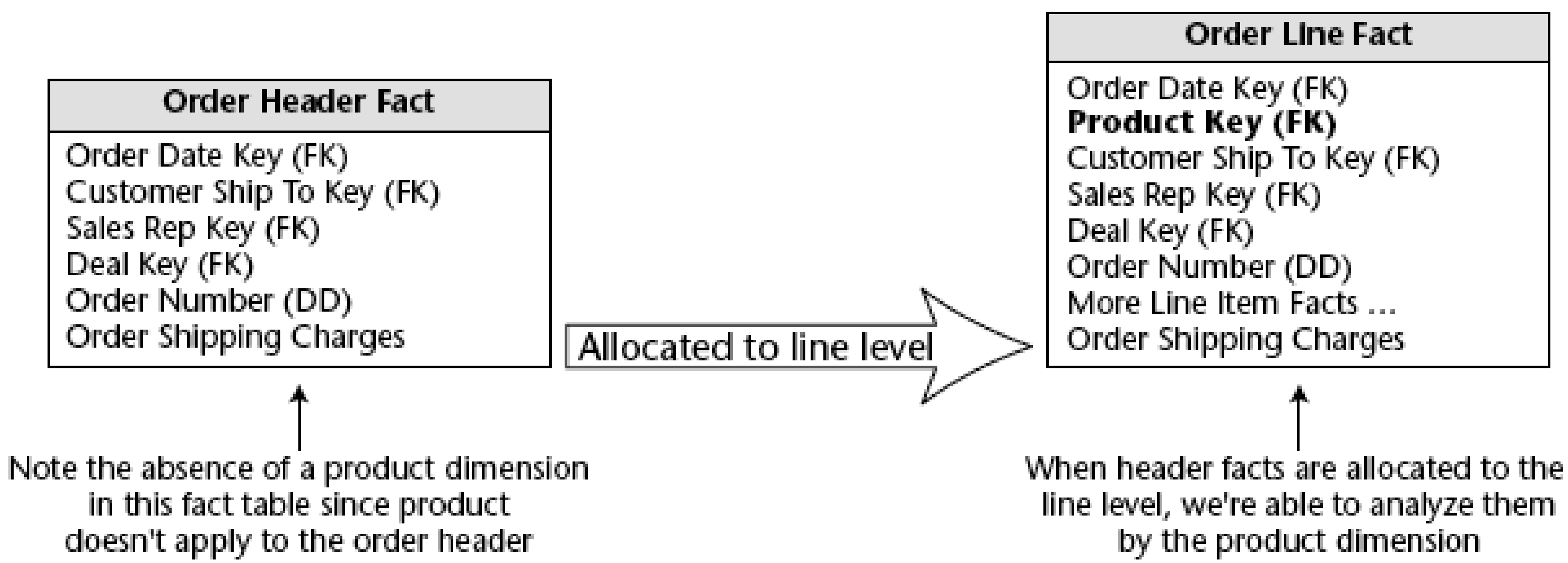
- The conversion rate table contains all combinations of effective currency exchange rates going in both directions because the symmetric rates between two currencies are not exactly equal.



[Kimball, 2002]

Header and Line Item Facts with Different Granularity

- It is quite common in parent-child transaction databases to encounter facts of differing granularity.
  - On an order, for example, there may be a **shipping charge that applies to the entire order** that isn't available at the individual product-level line item in the operational system.
- A procedure broadly referred to as **allocating**. Allocating the parent order facts to the child line-item level is critical if we want the ability to slice and dice and roll up all order facts by all dimensions, including product, which is a common requirement.



[Kimball, 2002]

Header and Line Item Facts with Different Granularity

- Without allocations, we'd be unable to explore header facts by product because the product isn't identified in a header-grain fact table. If we are successful in allocating facts down to the lowest level, the problem goes away.

> **We shouldn't mix fact granularities (for example, order and order line facts) within a single fact table. Instead, we need to either allocate the higher-level facts to a more detailed level or create two separate fact tables to handle the differently grained facts. Allocation is the preferred approach. Optimally, a finance or business team (not the data warehouse team) spearheads the allocation effort.**

[Kimball, 2002]

Fact Table Comparison

Fact Table Comparison

- **Transaction Fact Tables**: These fact tables represent an event that occurred at an instantaneous point in time.
- **Periodic Snapshot Fact Tables**: Periodic snapshots are needed to see the cumulative performance of the business at regular, predictable time intervals.
- **Accumulating Snapshot Fact Tables**: Represent an indeterminate time span, covering the complete life of a transaction or discrete product (or customer).

- These three fact table variations are not totally dissimilar because **they share conformed dimensions**, which are the keys to building separate fact tables that can be used together with common, consistent filters and labels. While the dimensions are shared, **the administration and rhythm of the three fact tables are quite different**.

[Kimball, 2002]

Fact Table Comparison

Characteristic	Transaction Grain	Periodic Snapshot Grain	Accumulating Snapshot Grain
Time Period Represented	Point in time	Regular predictable intervals	Inderterminate time span, typically short-lived
Grain	One row per transaction event	One row per period	One row per life
Fact table loads	Insert	Insert	Insert and update
Fact row updates	Not revisited	Not revisited	Revisited whenever activity
Date Dimension	Transaction Date	End-of period date	Multiple dates for standard milestones
Facts	Transaction activity	Performance for predefined time interval	Performance over finite lifetime

[Kimball, 2002]

Transaction Fact Tables

- The **most fundamental view of the business's operations** is at the individual transaction level.
- These fact tables represent an **event that occurred at an instantaneous point in time**.
- A row exists in the fact table for a given customer or product **only if a transaction event occurred**.

- The lowest-level data is the most naturally dimensional data, supporting analyses that cannot be done on summarized data. Transaction-level data let us analyze behavior in extreme detail.
- Once a transaction has been posted, we typically don't revisit it.

[Kimball, 2002]

Periodic Snapshot Fact Tables

- Periodic snapshots are needed to see the **cumulative performance** of the business at regular, **predictable time intervals.**
- With the periodic snapshot, we take a **picture** (hence the snapshot terminology) of the activity at the **end of a day, week, or month**, then another picture at the end of the next period, and so on. The periodic snapshots are stacked consecutively into the fact table.
- The periodic snapshot **represents an aggregation of the transactional activity** that occurred **during a time period**.
- The periodic snapshot fact table often is the only place to easily retrieve a regular, predictable, **view of trends of the key business performance metrics**.

[Kimball, 2002]

Accumulating Snapshot Fact Tables

- Accumulating snapshots almost always have **multiple date stamps**, representing the predictable **major events** or phases that take place during the course of a lifetime.
- Often there's an **additional date column that indicates when the snapshot row was last updated**.
- Since many of these dates are not known when the fact row is first loaded, we must use surrogate date keys to handle undefined dates.
- We **revisit** accumulating snapshot fact table rows to **update** them.
- Sometimes accumulating and periodic snapshots work in conjunction with one another.
  - When we build the monthly snapshot incrementally by adding the effect of each day's transactions to an accumulating snapshot.

[Kimball, 2002]

Designing Real-Time Partitions

Designing Real-Time Partitions

- The data warehouse now must **extend its existing historical time series** seamlessly **right up to the current instant**.
  - If the customer has placed an **order in the last hour**, we need to see **this order in the context of the entire customer relationship**.
  - Furthermore, we need to **track the hourly status of this most current order** as it changes during the day.

- Even though the gap between the operational transaction-processing systems and the data warehouse has shrunk in most cases to 24 hours, the rapacious needs of our marketing users require the data warehouse to fill this gap with near real-time data.

[Kimball, 2002]

Requirements for the Real-Time Partition

- The real-time partition is **a separate table subject to special update and query rules:**
  - **Contain all the activity that occurred since the last update of the static data warehouse**. We will assume that the static tables are updated each night at midnight.
  - **Link** as seamlessly as possible **to the grain and content of the static data warehouse fact tables**.
  - Be so **lightly indexed** that incoming data can be continuously dribbled in.

- The realtime partition has a **different structure corresponding to each fact table type**: **transaction grain**, **periodic snapshot grain**, and **accumulating snapshot grain**.

[Kimball, 2002]

Transaction Grain Real-Time Partition

- The real-time partition **has exactly the same dimensional structure** as its underlying static fact table.
  - It only contains the **transactions that have occurred since midnight**, when we loaded the regular data warehouse tables.
  - The real-time partition **may be completely unindexed** both because we need to maintain a **continuously open window for loading** and because there is no time series (since we only keep today's data in this table)
  - We **avoid building aggregates** on this table because we want a minimalist administrative scenario during the day.

- We attach the real-time partition to our existing applications by drilling across from the static fact table to the real-time partition. **Time-series aggregations** (for example, all sales for the current month) **will need to send identical queries to the two fact tables and add them together**.

[Kimball, 2002]

Transaction Grain Real-Time Partition

- <u>Example</u>: In a relatively large retail environment experiencing **10 million transactions per day**, the static fact table would be pretty big:
  - Assuming that each transaction grain record is 40 bytes wide (7 dimensions plus 3 facts, all packed into 4-byte fields), we accumulate 400 MB of data each day.
  - Over a year this would amount to about 150 GB of raw data. Such a fact table would be heavily indexed and supported by aggregates.

- The **daily tranche of 400 MB for the real-time partition could be pinned in memory**. Forget indexes, except for a B-Tree index on the fact table primary key to facilitate the most efficient loading.
- We send ***identical queries*** to the **static fact table** and the **real-time partition**

[Kimball, 2002]

Periodic Snapshot Real-Time Partition

- If the **static** data warehouse **fact table** has a **periodic grain** (say, monthly), then the **real-time partition can be viewed as the current hot-rolling month**.
  - Suppose that we are working for a big retail bank with 15 million accounts.
    - The static fact table has the grain of account by month. A 36-month time series would result in 540 million fact table records. Again, this table would be indexed extensively and supported by aggregates to provide good query performance.
  - The **real-time partition**, on the other hand, **is just an image of the current developing month**, updated continuously as the month progresses.
    - Semi-additive balances and fully additive facts are adjusted as frequently as they are reported. In a retail bank, the core fact table spanning all account types is likely to be quite narrow, of 480 MB that can be pinned in memory.

[Kimball, 2002]

Periodic Snapshot Real-Time Partition

- Query applications drilling across from the static fact table to the real-time partition have a slightly different logic compared with the transaction grain:
  - **Account balances** and other **measures of intensity** can be trended directly across the tables.
  - **Additive totals accumulated** during the current rolling period may **need to be scaled upward to the equivalent of a full month** to keep the results from looking anomalous.
  - Finally, on the last day of the month, hopefully the **accumulating real-time partition can just be loaded** onto the static data warehouse as the most current month, and the process can **start again with an empty real-time partition**.

[Kimball, 2002]

Accumulating Snapshot Real-Time Partition

- The real-time partition will consist of **only those line items which have been updated today**.
  - At the end of the day, the records in the real-time partition **will be** precisely **the new versions of the records that need to be written onto the main fact table** either by inserting the records if they are completely new or overwriting existing records with the same primary keys.
  - Typically, the realtime partition will be significantly smaller than in the first two cases and it will fit in memory.
  - Queries against an accumulating snapshot with a real-time partition need to fetch the appropriate line items from both the main fact table and the partition and can either drill across the two tables by performing a sort merge (outer join) on the identical row headers or perform a union of the rows from the two tables, presenting the static view augmented with occasional supplemental rows in the report.

[Kimball, 2002]

Further Reading and Summary

What you should know

- The differences between the three types of transaction tables and its main concerns
- The following concepts: Dimension Role-Playing; Product Dimension Revisited; Customer Ship-To Dimension; Deal Dimension; Degenerate Dimension for Order Number; Junk Dimensions; Multiple Currencies; Header and Line Item Facts with Different Granularity
- The need for real-time partitions and the proposed approach

[Kimball, 2002]

Further Readings

- **Further Readings**
  - The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition), Ralph Kimball, Margy Ross. 2002
    - From page 107 to 139

[Kimball, 2002]