

Armazéns de Dados

Departamento de Engenharia Informática (DEI/ISEP)
Paulo Oliveira
pjo@isep.ipp.pt

Adaptado do Original de:
Fátima Rodrigues (DEI/ISEP)

1

Bibliography

- Mastering the Data Warehouse Design
Relational and Dimensional Techniques
Claudia Imhoff, Nicholas Galemno, Jonathan G. Geiger
Wiley, 2003, First Edition
Chapters 2, 3, 4, 5 (pages 29-144)
- "From Enterprise Models to Dimensional Models: A Methodology for
Data Warehouse and Data Mart Design"
Daniel L. Moody, Mark A. R. Kortink
- The Data Warehouse Toolkit: The Complete Guide to Dimensional
Modeling
Ralph Kimball, Margy Ross
Wiley, 2002, Second Edition
Chapters 1,...,6

2

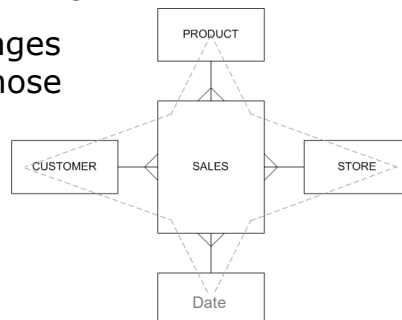
2

Brief Introduction to Dimensional Data Modeling

3

Dimensional Modeling

- Data modeling for DW is also known as **star schema creation**
- Based on **fact tables** and **dimension tables**
- Dimensional model is very **asymmetric**
 - one large dominant table in the center of the schema: **fact table**, with multiple joins connecting to **dimension tables**
- **Dimensional model** packages the data in a format whose design goals are:
 - **User understandability**
 - **Query performance**
 - **Flexibility to change**



4

4

Dimensional Modeling

- **Model simplicity** achieved by the reduced number of tables
 - Business users benefit because **data is easier to understand** and **navigate**
- Has also **performance benefits**
- Dimensional models are gracefully extensible to **accommodate change**
 - Can accommodate **new dimensions** as long as a single value of that dimension is defined for each existing fact row
 - Can accommodate **new facts** to the fact table, when the level of detail is consistent with the existing fact table
 - Can supplement preexisting dimension tables with **new attributes**

5

5

Fact Table

- **Primary table** in a dimensional model
- Holds
 - **Primary key made up by the foreign keys** (composite primary key) that connects to dimension tables
 - **Numerical measurements of the business** (the facts), which may be analyzed using statistical functions
 - ♦ units_sold, value_sold, order_cost, units_ordered, ...

6

6

Dimension Tables

- Provide the **basis for analyzing data** in the fact table
- Used to answer **"WHO"**, **"WHAT"**, **"WHEN"**, **"WHERE"** and **"WHY"** questions about the business events stored in the fact table
- Each dimension table has a **non-composite primary key** that usually corresponds to one of the components of the composite key in the fact table

7

7

Fact and Dimension Tables

- There is a **1:N** relationship between each **dimension table** and the **fact table**
- **Fact tables** typically have large volumes of rows, while **dimension tables** have a smaller number
 - Key advantage: **JOIN performance is improved** when one large table can be joined with some small tables
 - In many cases the **dimension tables** are small enough to **be fully cached in memory**

8

8

Schema Types

■ Star schema

- Fact table in the middle connected to a set of dimension tables

■ Snowflake schema

- Refinement of star schema where some dimensional table is normalized into a set of smaller dimension tables, forming a snowflake

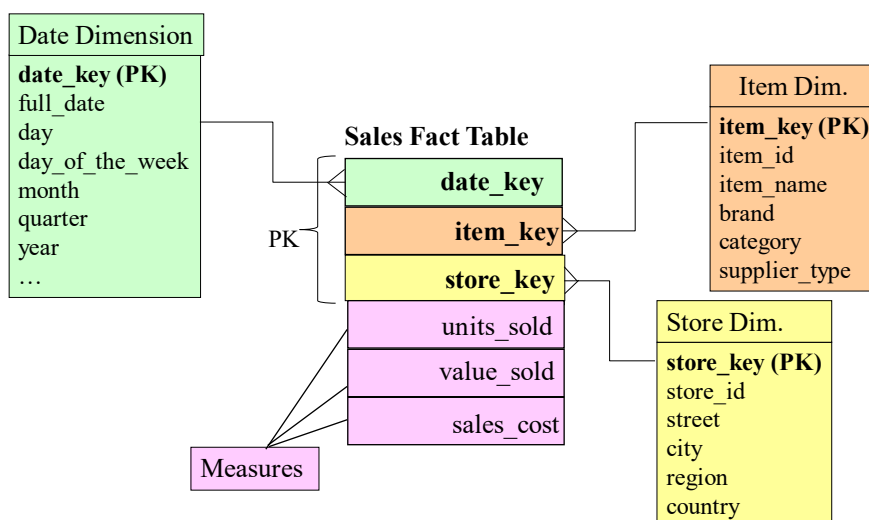
■ Galaxy schema/Fact constellation

- Multiple fact tables share dimension tables, viewed as a collection of star schemas

9

9

Example of a Star Schema



10

10

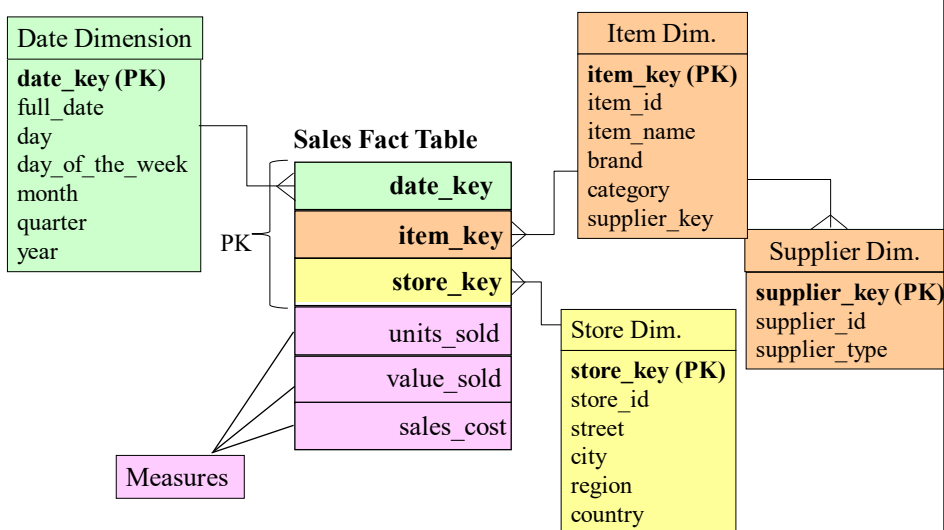
Date Dimension Example

date_key	fulldate	day	day_of_week	month	quarter	year	...
...
727	28/12/2022	28	wednesday	12	4	2022	...
728	29/12/2022	29	thursday	12	4	2022	...
729	30/12/2022	30	friday	12	4	2022	...
730	31/12/2022	31	saturday	12	4	2022	...
731	01/01/2023	1	sunday	1	1	2023	...
...

11

11

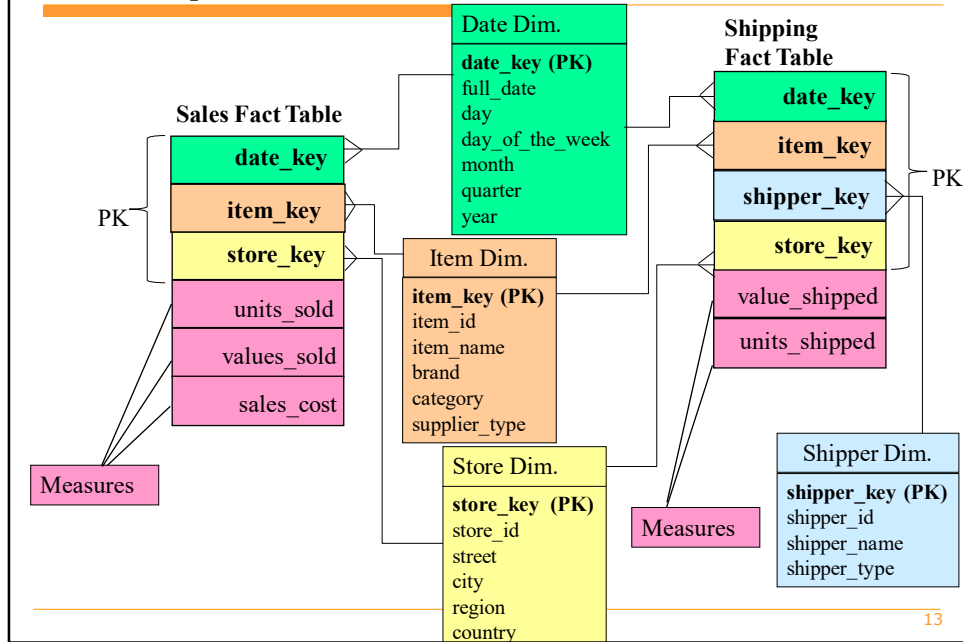
Example of a Snowflake Schema



12

12

Example of Fact Constellation



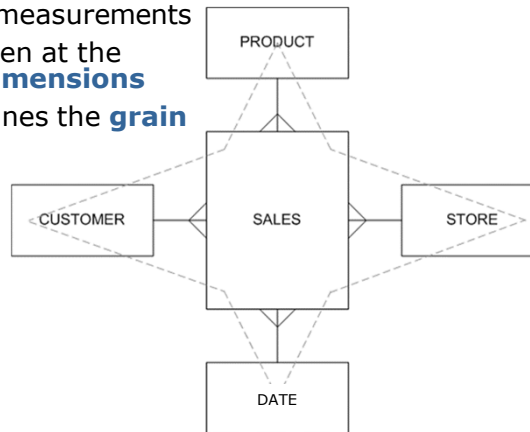
13

Dimensional Data Modeling

14

Dimensional Modeling

- Build around the **numerical measurements** of the business
 - Fact tables contain **measurements**
 - Dimension tables contain the **context** surrounding measurements
 - Measurements are taken at the **intersection of all dimensions**
 - List of dimensions defines the **grain of the fact table**



4

15

Fact Table

- Is the **primary table** in a dimensional model
- Holds the **measurements** of the business
- Composed by a **set of foreign keys** that connect to the dimension tables
- Its primary key is made up by **the set or a subset of the foreign keys**
- Role of a **normalized n-ary associative entity**
- All measurements in a fact table must be at the **same grain**

16

16

Importance of the Granularity Level

- **Business perspective** – it dictates the potential capability and flexibility of the DW
 - It is **impossible** to **answer questions** that **require details below the adopted level**
- **Technical perspective** – is one of the **major determinants of the DW size**
 - Has a significant impact on its **operating costs** and **performance**
- **Project perspective** – affects the amount of work that the project team will need to perform to create the DW
 - As the granularity level gets into greater levels of detail, the project team **needs to deal with more attributes and their relationships**

17

17

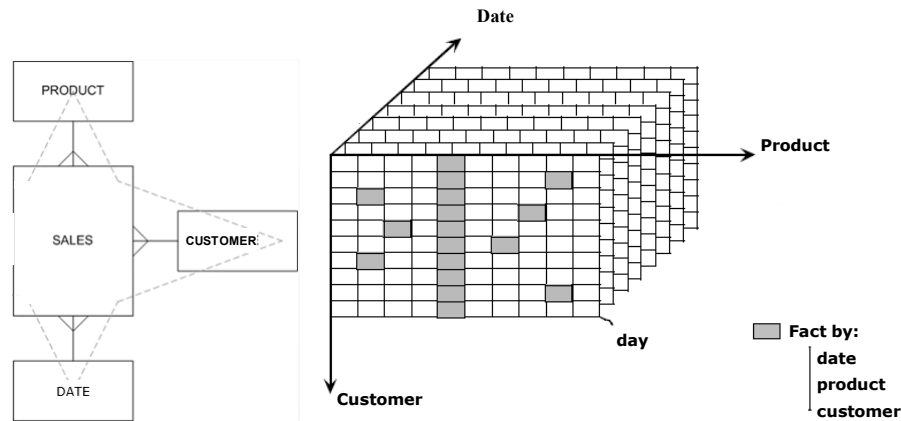
Fact Table doesn't store "non-events"

- Very important **not to try** to fill the fact table with zeros representing "nothing happened"
 - If there is no sales activity on a given day, in a given store, for a given product, the record must be left out of the fact table
- By only including true activity, **fact tables** tend to be **quite sparse**
- Despite their sparsity, **fact tables usually make up 90% or more of the total space consumed** by a dimensional database

18

18

Sparsely Fact Table



19

19

Kind of Facts

- **Additive facts**

Best and most useful facts because DW applications bring thousands, or even millions of fact rows at a time, and the most useful thing to do is to add them

- **Semi-additive facts**

Can be added only along some of the dimensions

- **Non-additive facts**

Simply cannot be added at all – can be counted or averaged

20

20

Kind of Facts

- **Additive facts**

Product Sales			
Product	Day 1	Day 2	Product Total
P1	200 €	300 €	500 €
P2	30 €	20 €	50 €
P3	100 €	300 €	400 €
P4	40 €	60 €	100 €
P5	300 €	100 €	400 €
Day Total	670 €	780 €	1 450 €

- **Semi-additive facts**

Customer Balance			
Customer	Day 1	Day 2	Customer Total
C1	200 €	300 €	500 €
C2	300 €	100 €	400 €
C3	100 €	100 €	200 €
C4	400 €	200 €	600 €
C5	30 €	40 €	70 €
Day Total	1 030 €	740 €	

- **Non-additive facts**

Sales Margin (sales profit / value sold)			
Product	Day 1	Day 2	Product Total
P1	33%	35%	68%
P2	27%	27%	54%
P3	45%	43%	88%
P4	17%	16%	33%
P5	8%	12%	20%
Day Total	130%	133%	263%

21

21

Dimension Tables

- Define the details of each transaction
- Dimension tables answer the “**who**”, “**what**”, “**when**”, “**where**” and “**why**” of a business event
 - For example, a sales transaction may be defined by a number of components:
 - ♦ Customer: **who** made the purchase
 - ♦ Product: **what** was sold
 - ♦ Store: **where** it was sold
 - ♦ Date: **when** it was sold
 - ♦ Promotion: **why** it was sold

22

22

Dimension Tables

- **Provide the context for fact tables**, that is, the context for all the **measures**
- Dimension tables have many columns or attributes
 - Usual for a dimension to have between 50 to 100 attributes
 - Relatively small in terms of the number of rows
 - Usually **much smaller than fact tables**
- Best attributes are **textual** and **discrete**
- **Entry points** into the fact table

23

23

Dimension Attributes

- Serve as the primary source of **query constraints**, **groupings**, and **report labels**
- Key to making the DW usable and understandable – **DW is only as good as the dimension attributes**
- Each dimension is defined by its **single primary key – surrogate key**, which serves as the basis for referential integrity with the fact table(s) to which it is joined

24

24

Surrogate Keys

- Joins between dimensions and fact tables should be based on **meaningless integer surrogate keys**
 - Other names: integer keys, *no natural* keys, artificial keys, synthetic keys
- Must be assigned **sequentially**
- **Benefits:**
 - Performance advantages
 - Protects the DW from operational changes
 - Allow the integration of data from multiple operational source systems
 - Allow to define artificial keys without meaning in operational systems, like "not applicable", "no effect", ...

25

25

Dimension Tables - Hierarchies

- Dimension tables typically are **highly denormalized**
- Often contain **hierarchical relationships** in the business
 - Example: Product dimension table – products roll up into subcategories and then into categories
- Hierarchical descriptive information is **stored redundantly**, to promote **ease of use and query performance**
- Usually small
 - < 10% of total data storage requirements
 - Since dimension tables are geometrically smaller than fact tables, **improving storage efficiency by snow-flaking has virtually no impact on the overall database size**

26

26

The Grocery Store

27

Grocery Store Business - Brief Description

- The business has **500 large grocery stores** spread over the country. Each store is divided by **departments** such as grocery, frozen foods, dairy, meat, bakery, floral, drugs,... Each store has roughly **60000 individual products** (called **Stock Keeping Units – SKUs**) on its shelves. About 40000 SKUs come from outside manufactures and have bar codes called **Universal Product Codes – UPCs**.
- The remaining 20000 SKUs come from departments like the meat, bakery, or floral departments and don't have UPC codes. Nevertheless, as a grocery store, these products **also have SKU numbers** assigned to them.
- At the grocery store, management is concerned with the sales of the products as well as maximizing the profit at each store. The most significant management decisions have to do with **pricing, promotions** and **good visibility of promotions**.

28

28

Kimball Dimensional Modeling Steps

1. Identify the business process

- Business process is a major operational process supported by some computational system(s) from which data can be collected for the purpose of data warehousing (e.g.: orders)

2. Identify the level of detail (grain)

- Detail level of the data to be represented in the fact table
- Determines the dimensionality of the underlying database and has a profound impact on its size

3. Identify the dimensions

- Choose the dimensions that will apply to each fact table
- For each dimension describe all its attributes

4. Identify the facts

- Choose the measures that will populate each fact table record

29

29

Modelling Grocery Store Business

1. Business process

- Sales

2. Granularity level (level of detail)

Options:

- Sales of products by store by promotion and by individual customer ticket transaction
 - ➔ In this grocery store chain, there is no effective way of identifying individual customers at the cash register
- Sales of products by store by promotion and by day (or by week or by month)
 - ➔ Weekly or monthly storage item movement would miss too many important analysis, such as difference in sales between Mondays and Saturdays

Best grain for this grocery store chain DW is considered to be the **product (or SKU) sales, by store, by promotion and by day**

30

30

Modelling Grocery Store Business

3. Dimensions involved

- Date
- Product
- Store
- Promotion

4. Facts/Measures of interest

- Value sold
- Units sold
- Sales cost
- Sales profit
- Sales margin

31

31

Date Dimension

- **Date dimension** is present in every DW, because every DW is a time series

Date Dimension
date-key
full-date
day-week
day-number-month
day-number-year
week-number
month-name
month-number
semester
quarter
year
last-day-month-flag
season
...

Unlike almost all the other dimensions, **date dimension can be build in advance** – five or ten year of history records can be loaded, as well the next few years

- Surrogate key assigned to the date dimension **should be assigned consecutively in the order of date**
- It's the only dimension where the **surrogate key has a relationship with the underlying semantics** of the dimension

32

32

Time of Day as a Dimension or Fact

- **Time (of day) dimension** has one row per discrete time period necessary to the business
 - Example: each second or minute within a 24-hour period
- **Time (of day) must be separated from the date dimension** to avoid an explosion in the row count
- **Preferred modeling design if we need to support the roll-up of time periods into more summarized groupings** for reporting and analysis, such as 15-minute interval, half-hours, hours, ...
- If there is no need to roll up/drill down or filter on time-of-day groups, then **time can be treated as a simple numeric fact**
 - Time of day can be expressed as the number of minutes or seconds since 00:00

33

33

Separate Date and Time Dimensions

date_key	fulldate	day	day_of_week	month	quarter	year	...
...
727	28/12/2022	28	wednesday	12	4	2022	...
728	29/12/2022	29	thursday	12	4	2022	...
729	30/12/2022	30	friday	12	4	2022	...
730	31/12/2022	31	saturday	12	4	2022	...
...

time_key	fulltime	hour	minute	second	period	...
...
1527	11:03:20	11	03	20	Morning	...
1528	11:03:21	11	03	21	Morning	...
1529	11:03:22	11	03	22	Morning	...
1530	11:03:23	11	03	23	Morning	...
...

34

34

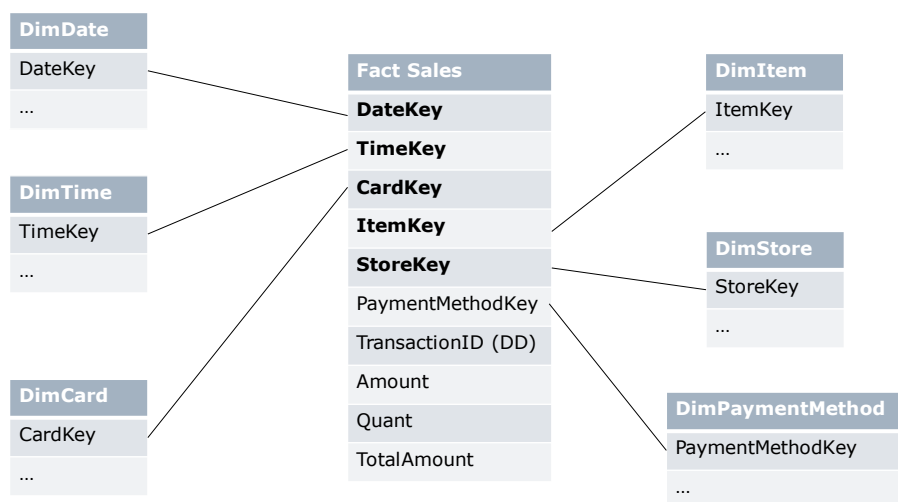
Separate Date and Time Dimensions

- Nr. of Records for Date Dimension
 - Granularity: day
 - Period: 20 years
 - $365 * 20 + 5$ (leap years) = 7 305 records
- Nr. of Records for Time Dimension
 - Granularity: second
 - Period: 24 hours
 - $60 * 60 * 24 = 86\,400$ records

35

35

Separate Date and Time Dimensions



36

36

Single Date and Time Dimension

datetime_key	fulldate	day	day_of_week	fulltime	hour	minute	second
...
17274	28/12/2022	28	wednesday	11:03:20	11	03	20
17275	28/12/2022	28	wednesday	11:03:21	11	03	21
17276	28/12/2022	28	wednesday	11:03:22	11	03	22
17277	28/12/2022	28	wednesday	11:03:23	11	03	23
17278	28/12/2022	28	wednesday	11:03:24	11	03	24
17279	28/12/2022	28	wednesday	11:03:25	11	03	25
17280	28/12/2022	28	wednesday	11:03:26	11	03	26
17281	28/12/2022	28	wednesday	11:03:27	11	03	27
...

37

37

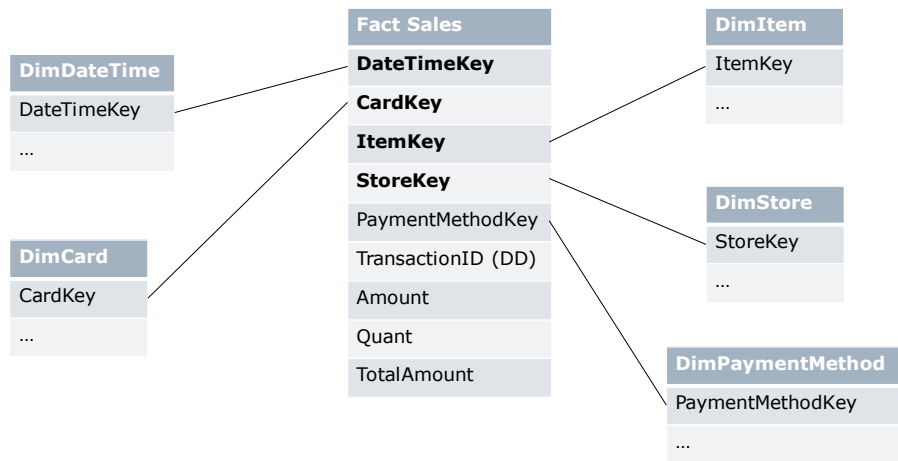
Single Date and Time Dimension

- Nr. of Records for Date Dimension
 - Granularity: day second
 - Period: 24 hours of 20 years
 - $(365 * 20 + 5) * (60 * 60 * 24) =$
 $7\,305 * 86\,400 = 631\,152\,000$ records

38

38

Single Date and Time Dimension



39

39

Product Dimension

Product dimension describes every SKU with as many descriptive attributes as possible, including the **existing hierarchies**

Product Dimension
product-key
SKU-description
SKU-number
package-size
brand
subcategory
category
department
package-type
diet-type
weight
weight-unit
...

It is possible to **browse** among dimension attributes **whether or not they belong to a hierarchy** and it is possible to **roll up** and **drill down** using the attributes that **belong to a hierarchy**

40

40

Store Dimension

Store dimension describes every store in the grocery chain –
geographic dimension

Store Dimension
store-key
store-name
store-number
store-address
store-zip
store-city
store-district
store-region
store-manager
open-date
last-remodel-date
store-sqft
grocery-sqft
...

Numeric attributes,
however they are clearly a
constant attribute of store

41

41

Promotion Dimension

- **Promotion dimension** – describes each promotion condition under which a product is sold in the grocery chain
- **Causal dimension** – describes factors that cause a change in product sales
- Needs a **special register "N/A"** to join sales in fact table without promotion

Promotion Dimension
promotion-key
promotion-name
price-reduction-type
ad-type
display-type
coupon-type
ad-media-name
display-provider
promo-cost
promo-begin-date
promo-end-date
...

42

42

Fact Table

Sales Fact
date-key
product-key
store-key
promotion-key
value-sold
units-sold
sales-cost
sales-profit

date-key	...	value-sold	units-sold	sales-costs	sales-profit	sales-margin
101	...	780	78	263	517	0,66
102	...	1044	18	580	464	0,44
103	...	213	10	140	73	0,34
104	...	95	19	39	56	0,59
Total		2132	125	1022	1110	0,52

Business Measures / Facts

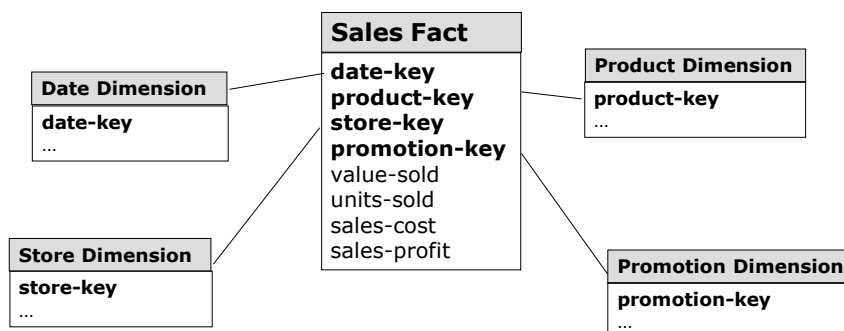
Is not stored in the Fact Table

- First three facts are **additive**
- sales-profit** = value-sold – sales-cost → **additive**
- sales-margin** = sales-profit / value-sold
 → **No-additive calculation** – can be calculated for any slice of fact table by calculating the sales profit and value sold before dividing

43

43

Grocery Store Business Schema



Advantages:

- Easy to understand
- Better performance
- Easy extensible: new dimensions and new facts

44

44

Typical Query on a Star Schema

- **Sales** for the **year of 2019** of the **store "Oxford Street"** by **brand**

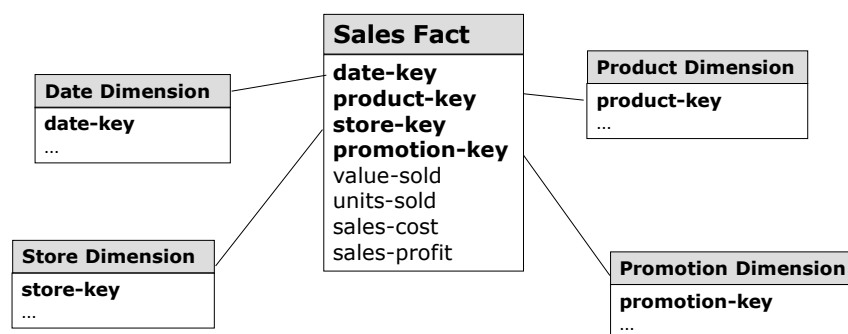
```
SELECT p.brand, sum(f.value-sold), sum(f.units-sold)
FROM Product p, Date d, Store s, SalesFact f
WHERE f.product-key = p.product-key
      AND f.store-key = s.store-key
      AND f.date-key = d.date-key
      AND d.year = 2019
      AND s.store-name = "Oxford Street"
GROUP BY p.brand
```

- First, **constraints (slicing)** are processed for each dimension
- Each dimension produce **a set of candidate keys**
- Candidate keys **are combined (by Cartesian product) to build the composite keys to be searched in the fact table**
- All rows in the fact table are **selected, grouped and aggregated according to the query**

45

45

Grocery Store Business Schema



- Grocery store star schema cannot answer an important question:
 - **What products were on promotion but didn't sell ?**

46

46

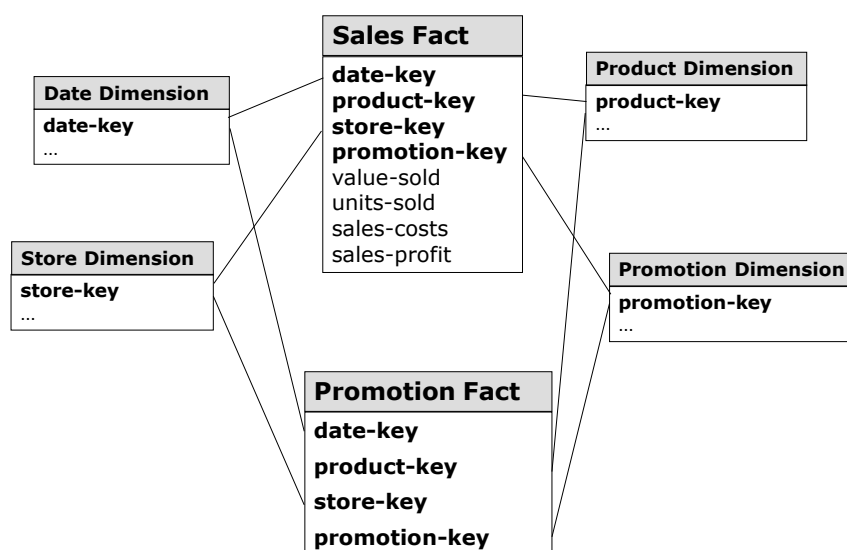
Factless Fact Table

- Sales fact table **only records products sold** – there are no fact table rows with zero facts
- To know which products on promotion that didn't sell is necessary a **second fact table promotion**
- Promotion coverage fact table
 - Keys: date, product, store, and promotion
 - Grain: **one row for each product on promotion in a store each day**
 - Measures: no measures
 - ♦ This kind of fact table without metrics is called **Factless Fact Table**

47

47

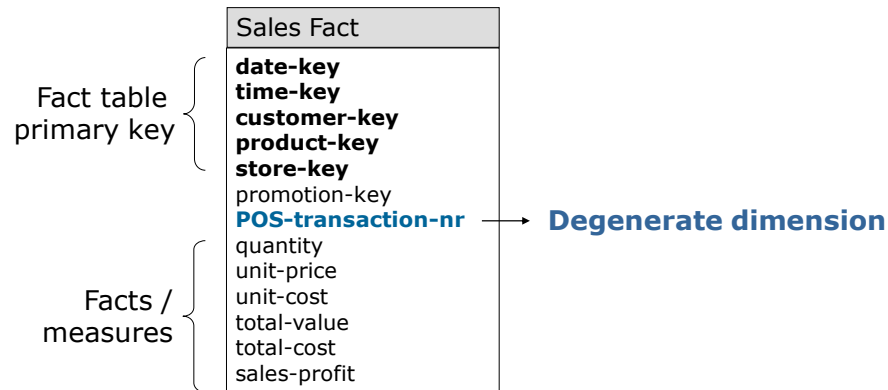
Factless Fact Table (Promotion Fact)



48

48

Degenerate Dimension



- Document control numbers such as order numbers, invoice-numbers,... are represented as **degenerate dimensions** – **dimension keys with no corresponding dimension tables**
- Provide a **direct reference to the operational system**

49

49

Dimensional Model for Inventory Periodic Snapshot

50

Modelling Inventory Periodic Snapshot

1. Business process

- Inventory
 - ♦ Optimized inventory levels can have an impact on profitability
 - ♦ Making sure the **right product** is in the **right store** at the **right time** minimizes out-of-stocks and reduces overall inventory carrying costs

2. Granularity

- Daily quantity on stock by product and store

3. Dimensions

- Date
- Product
- Store

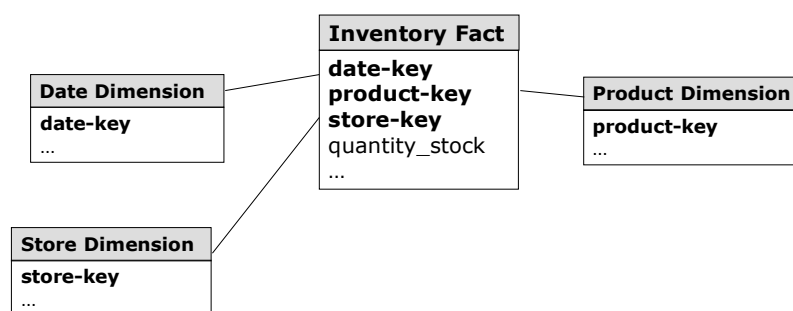
4. Facts

- Quantity on stock
- ...

51

51

Inventory Periodic Snapshot Schema



- Dimensions **are the same** as on the grocery store schema
- Dimension tables **must have additional attributes useful for inventory analysis**, such as:
 - Product dimension: Minimum reorder quantity
 - Store dimension: Frozen and refrigerated storage square footages

52

52

Fact Table Sizing

- Inventory fact tables are **very dense** by nature – every product must be represented in every store, every day
 - 60000 products × 500 stores × 1095 days (3 years) = 32,85 billion records
 - Size of fact record: 8 bytes
 - Fact table size = 32,85 billion records × 8 bytes = **244,75 GB**
- Dense inventory databases **require some compromises**
 - Keep the last 30 days of inventory at the daily level
 - Revert to weekly inventory the most recent 11 months
 - Revert to monthly inventory snapshots the prior two years
 - Instead of 1095 time snapshots, the number is reduced to:
 - ♦ 30 + 48 + 24 = 102 snapshots, which **reduces database size more than a factor of 10**

53

53

Semi-additive Facts

Store	S1	S2
	2 un. P1	3 un. P1
	3 un. P2	1 un. P2
Day1	1 un. P3	3 un. P3
	4 un. P4	2 un. P4
	3 un. P5	4 un. P5
	1 un. P1	2 un. P2
Day2	3 un. P2	1 un. P2
	4 un. P3	5 un. P3
	2 un. P4	0 un. P4
	1 un. P5	1 un. P5

- Given a product and a date the **quantity on stock** can be summed-up by stores
- Given a product and a store **it is not possible to sum-up by date**
- Possible aggregations on store S1 for product P4:
 - Average = $(4+2)/2=3$ → **average over nr. of periods**
 - Min=2
 - Max=4

54

54

Semi-additive Facts

All measures that represent snapshots of a level or balance at one point in time (inventory levels, financial account balances, and measures such as weights) are **inherently non-additive across the date (and time) dimension(s)** – in these cases, the measure may be aggregated usefully across date (and time), for example, **by averaging over the number of periods**

55

55

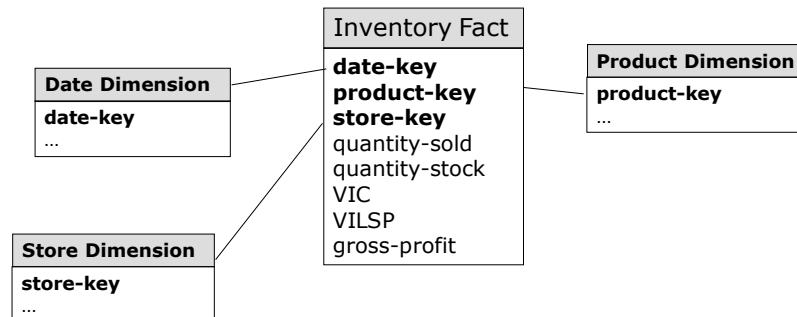
Enhanced Inventory Facts

- **Quantity on stock** needs to be used in conjunction with additional facts to develop **other metrics**
 - Quantity sold
 - Stock financial value
 - ♦ Value of the Inventory Cost (VIC)
 - $VIC = \text{Stock Quantity} * \text{Unit Cost}$
 - ♦ Value of the Inventory at Latest Selling Price (VILSP)
 - $VILSP = \text{Stock Quantity} * \text{Unit Price}$
 - With previous measures is possible to calculate:
 - ♦ $\text{Gross Profit} = VILSP - VIC$
 - ♦ $\text{Gross Margin} = \text{Gross Profit} / VILSP$

56

56

Enhanced Inventory Periodic Snapshot Schema



- Quantity-sold is **additive** across all dimensions
- Measures quantity-stock, VIC, VILSP, and gross-profit **are semi-additive**
- Gross margin is not stored in the fact table because it is **non-additive**

57

57

Slowly Changing Dimensions

58

Dimension Characteristics

- Up to this point we have assumed that dimensions are **independent of time**
- While **dimension table attributes are relatively static** in the real world, **their values are not fixed forever**
- **Dimension attributes can change slowly over time**
 - Products change size and weight
 - Customers relocate
 - Stores change layouts
 - Sales staff are assigned to different locations
- In a DW **it is necessary to know the history of values to match the history of facts** with the **correct dimensional records** at the time the facts happened

59

59

Slowly Changing Dimension Techniques

- For **each attribute** in dimension tables, it is necessary a **strategy to handle change**
- When an **attribute value changes** in the operational system, the **five strategies most common** to respond to that change are:
 - **Type 1**: Overwrite the value
 - **Type 2**: Add a dimension row
 - **Type 3**: Add a dimension column
 - **Type 4**: New history table
 - **Type 6**: Hybrid approach
- Each strategy results in a **different degree of tracking changes** over time

60

60

Type 1: Overwrite the Value

- **Overwrites the old attribute value** in the dimension row, replacing it with the current value
 - Attribute always reflects the most recent assignment
 - No changes are needed elsewhere in the dimension record
 - No keys are affected anywhere in the DW
 - Easiest to implement
 - **It is impossible to track history**
- **Appropriate for corrections or** when there is **no interest in** keeping the **old value**
 - Example: consider an electronic retailer whose marketing person decides that a specific software product **must belong to the Strategy department instead of the Education department**

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z

61

61

Type 1: Overwrite the Value

Updated row:

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	ABC922-Z

- Product key is the dimension key
- SKU is a natural key – must remain inviolate
- No dimension or fact table keys were modified when the IntelliKidz's department changed
- Rows in the fact table still reference product key 12345, regardless of IntelliKidz's departmental location
- **Any previous aggregations on this attribute will change and should be recalculated**
- **No history is recorded**

62

62

Type 2: Add a Dimension Row

- **Create an additional dimension record at the time of the change with the new attribute value**
 - Segments history very accurately between the old description and the new description
- **Predominant technique** for supporting slowly changing dimensions
- **Represents prior history correctly**
- **Gracefully tracks as many dimension changes as required**
- **One downside is to accelerate dimension table growth**

63

63

Type 2: Add a Dimension Row

Product Key	Product Description	Department	SKU Number (Natural Key)	Effective Date	Expired Date
12345	IntelliKidz 1.0	Education	ABC922-Z	11/02/2019	19/09/2020
25984	IntelliKidz 1.0	Strategy	ABC922-Z	19/09/2020	

- Each of the separate surrogate keys identifies **a unique product attribute profile that was true for a span of time**
- In the fact table
 - Fact rows for IntelliKidz prior to the date of change, would reference product key 12345
 - When the product was moved to the Strategy department, fact rows reference product key 25984 to reflect the change
- Constrains
 - On the **department attribute**, precisely differentiate between the two product profiles
 - On the **product description**, brings the complete product history
 - On the count distinct **SKU number** natural key, brings the correct number of products

64

64

Type 2: Add a Dimension Row

- Includes an **effective and expired date stamp on a dimension row with type 2 changes**
 - Date stamp refers to the moment when the attribute values in the row **become valid** (or invalid in case of expiration date)
 - Effective and expiration date attributes **are needed** to know which surrogate key is valid when loading fact records
- **No need to update** existing aggregation tables

65

65

Type 3: Add a Dimension Column

- Sometimes it is useful the ability to **see fact data as if the change never occurred**
- Happens most frequently with sales force reorganizations
 - District boundaries have been redrawn, but some users still want the ability to see **today's sales** in terms of **yesterday's district lines** just to see how they would have done under the old organizational structure
 - For a few transitional months, there may be a desire to track history in terms of the new district and conversely to track new data in terms of old district
- Type 2 SCD **doesn't support** this requirement
- While type 2 strategy partitions history, **it does not allow to associate the new attribute value with old fact history, or vice versa**

66

66

Type 3: Add a Dimension Column

- New dimension row is not created, but rather **a new attribute is added** to capture the value change
- In the case of IntelliKidz
 - Product dimension table is altered to add a **prior department** attribute
 - Prior department attribute is populated with the existing department value (Education)
 - Department attribute is updated to reflect the new value (Strategy)

Product Key	Product Description	Department	Prior Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	Education	ABC922-Z

67

67

Type 3: Add a Dimension Column

- **All existing reports and queries use** the new department description immediately
 - Similar effect to Type 1 SCD
 - **Any previous aggregations must be recalculated**
 - Still possible to **analyse/report on the old department value** by querying on the **prior department** attribute
- Allows to see new and historical fact data by either the **new or prior attribute value**
- Appropriate when there is a strong need to **support two views of the world simultaneously**
- Although the change has occurred, it is still logically possible **to act as if it has not**
- **Inappropriate if it is necessary to track the impact of numerous intermediate attribute values** – in this case a type 2 response should be used

68

68

Type 4: New History Table

- Idea behind is **almost the same as SCD Type 2**
- Also maintains the historical data **but not in the same table**
- Creates a **new history table** to keep the historical data
- Makes dimensions smaller, reduces complexity and improves performance **if the majority of uses only need the current value**
- Provides a solution to handle **the rapid changes in the dimension tables**, that causes performance and maintenance issues when using SCD Type 2

69

69

Type 4: New History Table

First Time Load on 08-10-2020

Source Table

product_id	product_name	unit_price	product_desc	eligible_promotion	Sync
100	Prod1	100.5	Prod desc 1	1	0
101	Prod2	50	Prod desc 2	0	0



Target Table

product_key	product_id	product_name	unit_price	product_desc	eligible_promotion	start_date
1	100	Prod1	100.5	Prod desc 1	1	08-10-2020
2	101	Prod2	50	Prod desc 2	0	08-10-2020

Second Time Load on 09-10-2020

Source Table

product_id	product_name	unit_price	product_desc	eligible_promotion	Sync
100	Prod1	100.5	Prod desc 1	1	1
101	Prod2	35.5	Prod desc 2	1	0
102	Prod3	40.25	Prod desc 3	0	0



Target Table

product_key	product_id	product_name	unit_price	product_desc	eligible_promotion	start_date
1	100	Prod1	100.5	Prod desc 1	1	08-10-2020
3	101	Prod2	35.5	Prod desc 2	1	09-10-2020
4	102	Prod3	40.25	Prod desc 3	0	09-10-2020

Old/History Data



History Table

product_key	product_id	product_name	unit_price	product_desc	eligible_promotion	start_date	end_date
2	101	Prod2	50	Prod desc 2	0	08-10-2020	09-10-2020

70

70

Type 6: Hybrid Approach

- Hybrid approach that **combines the three fundamental SCD techniques**
- Characteristics
 - **Creates a new row** to capture the change (SCD Type 2)
 - **Has an attribute** to reflect an alternative view of the world (SCD Type 3)
 - **Values are overwritten** for all earlier dimension rows for a given change (SCD Type 1)
- **Type 6 = Type 2 + Type 3 + Type 1**
- Adds current attributes alongside the historical attributes so measures can be **grouped by the historical or current dimension attribute values**

71

71

Type 6: Hybrid Approach

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Historic Department Name	Current Department Name	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	Education	2012-01-01		Current

Rows in Product dimension following first department reassignment:

Product Key	SKU (NK)	Product Description	Historic Department Name	Current Department Name	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	Strategy	2012-01-01	2012-12-31	Expired
25984	ABC922-Z	IntelliKidz	Strategy	Strategy	2013-01-01		Current

Rows in Product dimension following second department reassignment:

Product Key	SKU (NK)	Product Description	Historic Department Name	Current Department Name	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	Critical Thinking	2012-01-01	2012-12-31	Expired
25984	ABC922-Z	IntelliKidz	Strategy	Critical Thinking	2013-01-01	2013-02-03	Expired
31726	ABC922-Z	IntelliKidz	Critical Thinking	Critical Thinking	2013-02-04		Current

72

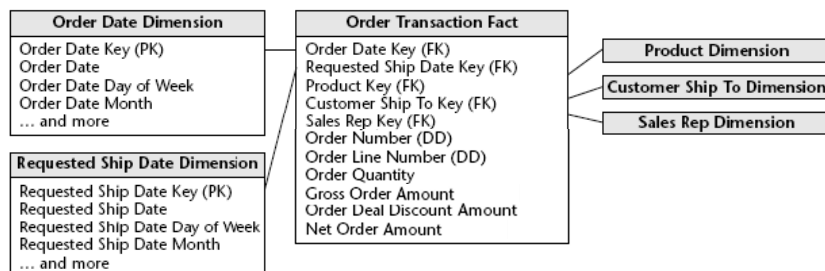
72

Order Management

73

Order Transactions

- **Natural granularity** for an order transaction fact table
 - One row for each line item of an order
- Facts associated with this process typically include
 - Order quantity
 - Order amount (gross and net)
 - Order discount amount



Order Transaction Dimensional Model

74

74

Dimension Role-Playing

- Many dates can be associated with each transaction
 - Order date
 - Requested ship date
 - Delivered date
- **Each of these dates should be a foreign key** in the fact table
- It is not possible to join the two foreign keys with the same date dimension – SQL would interpret such a **two-way simultaneous join** as requiring both dates to be equal

```
SELECT OrderID
FROM ORDERS, DIMDATE D
WHERE OrderDateKey = D.Date_Key AND ReqShippedDateKey = D.Date_Key

SELECT OrderID, D1.Month_Name, D2.Month_Name
FROM ORDERS, DIMDATE D1, DIMDATE D2
WHERE OrderDateKey = D1.Date_Key AND ReqShippedDateKey = D2.Date_Key
```

75

75

Dimension Role-Playing

- Create the illusion of **independent date tables** by using views
- Each of the SQL views must carefully **uniquely label the columns**

```
CREATE VIEW ORDER-DIMDATE (Order_Date_Key, Order_Day_Of_Week,
                           Order_Month, ...)
AS SELECT Date_Key, Day_Of_Week, Month_Name, ... FROM DIMDATE

CREATE VIEW REQ-SHIP-DIMDATE (Req_Ship_Date_Key, Req_Ship_Day_Of_Week,
                             Req_Ship_Month_Name, ...)
AS SELECT Date_Key, Day_Of_Week, Month_Name, ... FROM DIMDATE

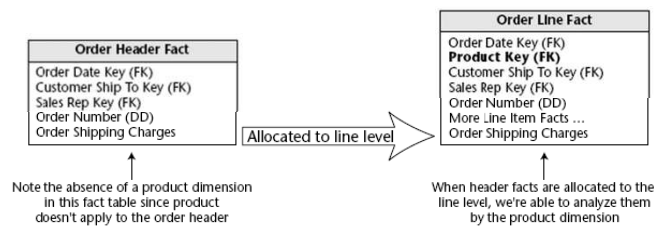
SELECT Order_ID, Order_Month, Req_Ship_Month_Name
FROM ORDERS, ORDER-DIMDATE, REQ-SHIP-DIMDATE
WHERE OrderDateKey = Order_Date_Key AND ReqShipDateKey = Req_Ship_Date_Key
```

76

76

Header & Line Facts with Dif. Granularities

- Quite common in parent-child transaction databases to **find facts with different granularity**
 - On an order there may be a **shipping charge that applies to the entire order** that isn't available at the individual product-level line
- To handle them it is necessary to make a procedure broadly referred to as **allocating**
 - Allocating the parent order facts to the child/item-line level **is critical** if the ability to drill-down and roll-up order facts by all dimensions is wanted

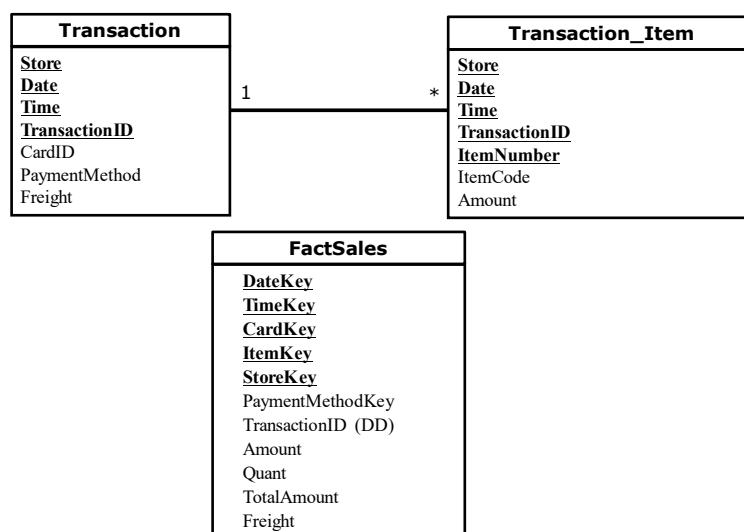


77

77

Header & Line Facts with Dif. Granularities

Example from Lab Classes



78

78

Header & Line Facts with Dif. Granularities

Transaction						
Store	Date	Time	TransactionID	CardID	PaymentMethod	Freight
S0002	20010719	1650	T10000261	C0100000111	CC	15

Transaction_Item						
Store	Date	Time	TransactionID	ItemNumber	ItemCode	Amount
S0002	20010719	1650	T10000261	1	I0000000492	59
S0002	20010719	1650	T10000261	2	I0000000486	49
S0002	20010719	1650	T10000261	3	I0000000262	189

FactSales							
DateKey	ProductKey	...	TransactionID	Amount	Quant	TotalAmount	Freight
542	232	...	T10000261	59	1	59	5
542	351	...	T10000261	49	1	49	5
542	142	...	T10000261	189	1	189	5

Agregating fact sales transaction and summing freight: 15

79

79

Junk Dimensions

- Within a complex operational system there are always a number of **miscellaneous indicators and flags**, each of which takes on a small range of discrete values
- There are many approaches to handle this kind of information
 - Leave the flags and indicators unchanged in the fact table row**
 - It's not possible to filter out based on the fact table attributes
 - Make each flag and indicator into its own separate dimension**
 - Could cause a 5-dimension design to move into a 25-dimension design
 - Strip out all the flags and indicators from the design**
 - Information is lost
 - Appropriate approach** is to study them carefully and then **pack them into one or more junk dimensions**

80

80

Junk Dimensions

- Is a convenient grouping of **typically low-cardinality flags and indicators**
- Flags and indicators are **removed from the fact table** and placed into a **useful abstract dimension**

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

81

81

Junk Dimension: Example From Lab Classes

- DimPaymentMethod

PaymentMethodKey	PaymentMethodID	PaymentMethodDescription
1	CA	Cash Advance
2	DC	Debit Card
3	CC	Credit Card
4	CH	Check

- DimStore

StoreKey	StoreID	StoreName
1	S0002	Princeton
2	S0006	Foxton
3	S0015	Ravensville
4	S0034	Catburg
5	S0040	Kingsville
6	S0056	Queensbury

82

82

Junk Dimension: Example From Lab Classes

- DimJunk

JunkKey	PaymentMethodID	StoreID	PaymentMethodDescription	StoreName
1	CA	S0002	Cash Advance	Princeton
2	CA	S0006	Cash Advance	Foxton
3	CA	S0015	Cash Advance	Ravensville
4	CA	S0034	Cash Advance	Catburg
5	CA	S0040	Cash Advance	Kingsville
6	CA	S0056	Cash Advance	Queensbury
7	DC	S0002	Debit Card	Princeton
8	DC	S0006	Debit Card	Foxton
9	DC	S0015	Debit Card	Ravensville
10	DC	S0034	Debit Card	Catburg
11	DC	S0040	Debit Card	Kingsville
12	DC	S0056	Debit Card	Queensbury
...
24	CH	S0056	Check	Queensbury

83

83

Multiple Currencies

- For a large multinational, order transactions are captured in different currencies
- One solution is to **include columns in the fact table for each currency**

FactOrders
<u>DateKey</u>
<u>CustomerKey</u>
...
UnitPrice_euros
UnitPrice_pounds
UnitPrice_dollars
TotalValue_euros
TotalValue_pounds
TotalValue_dollars
...

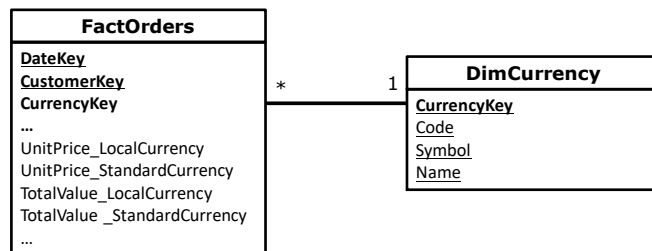
- However, there are an open-ended number of currencies

84

84

Multiple Currencies

- Another (better) solution
 - Order transactions expressed in both **local currency** and the **standardized corporate currency** (Euro, US Dollars, ...)
 - ♦ Each underlying order fact replaced by a pair of facts
 - ♦ Fact table supplemented with an **additional currency dimension** to identify the currency type associated with local-currency facts

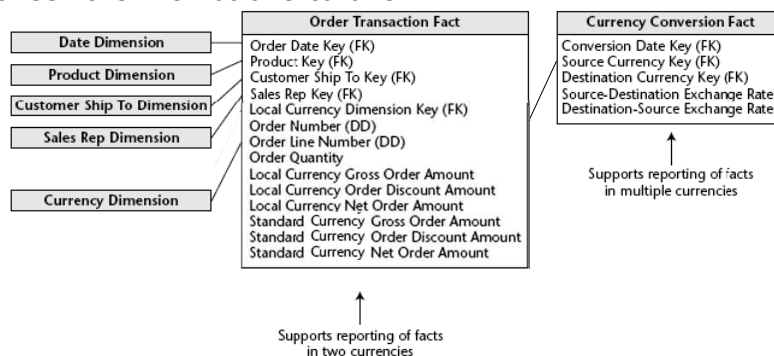


85

85

Currency Conversion Fact Table

- Most **flexible/powerful** solution
- Contains all combinations of currency exchange rates in both directions because **the symmetric rates between two currencies are not equal**
- **Represents currencies**, not countries, because the relationship between them is not one-to-one



86

86

Multiple Measure Units

- Sometimes different functional units within the organization want to see the **same performance metrics expressed in different measure units**
 - Manufacturing managers may want to see the product flow in terms of pallets or shipping cases
 - Sales and marketing managers, may wish to see the quantities in sales packs or consumer units
- Designers sometimes are tempted to **put the measure unit conversion factors** in the dimensions (such as a shipping case factor in the product dimension)
- **Conversion factors** to be applied to different units of measure **may change over time**, so users also would need to determine **which factor is applicable at a specific point in time**

87

87

Multiple Measure Units

- Rather than risk miscalculating the equivalent quantities by placing conversion factors in the dimension, **it is recommended that they are stored in the fact table**

Order Fulfillment Fact
Date Keys (FKs)
Product Key (FK)
More Foreign Keys ...
Degenerate Dimensions ...
Order Quantity
Release to Manufacturing Quantity
Manufacturing Pass Inspection Quantity
Manufacturing Fail Inspection Quantity
Finished Goods Inventory Quantity
Authorized to Sell Quantity
Shipment Quantity
Shipment Damage Quantity
Customer Return Quantity
Invoice Quantity
Retail Case Factor
Shipping Case Factor
Pallet Factor
Car Load Factor

The factors are physically packaged on each fact row. In the user interface, a view multiplies out the combinations

88

88

Multiple Measure Units

- In the orders fact table example, **10 basic fundamental quantity facts exist**, in addition to **5 units of measure**
- If physically stored all the facts expressed in the different units of measure, this results in **50 (10 x 5) facts in each fact row**
- Instead, by building an underlying physical row with 10 quantity facts and 4 unit-of measure conversion factors, the physical design now has **only 14 quantity-related facts (10 + 4)**
- **Converted facts are presented to users using a view**
- **Extra computation** involved in multiplying quantities by conversion factors **is negligible** compared with other DBMS operations

89

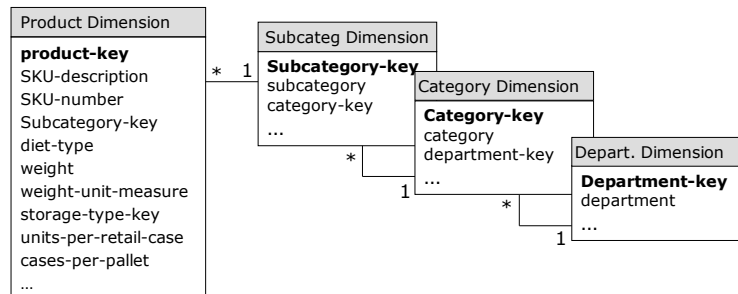
89

Big Dimensions (e.g., Product and Customer)

90

Resist to Snowflaking

Dimensions with hierarchies are sometimes decomposed by designers into a **snowflake structure**



Reason to snowflake is to save storage space, however:

Data mart size:	50 GB
Fact table size:	45 GB
Product dimension:	2,5 GB
Savings by snowflaking:	0,15 GB

Total database size, with or without snowflaking: 50 GB (rounded)

91

91

Snowflake vs. Query Performance

- If a large dimension has been **split into a snowflake**, and a data analysis demands separated attributes lying along various tables, **performance will be compromised**
- Best general purpose in a large dimension table is a **flat table implementation**, both to preserve the best **performance** and to present the **simplest view** to the end-user
- **Defendable exception** to snowflaking is when a **large dimension** has a **large block of frequently analysed demographic data** such as a customer dimension can have
 - Demographic data raises change-tracking challenges

92

92

Demographic Minidimension

- Most effective technique for using demographic attributes in a **large customer dimension** is to separate the frequently analysed or changing attributes into a **demographic minidimension**
- If the demographic variables are put into a separate table, **it is only necessary to store the distinct combinations of demographic attributes** that occur
 - For continuously attributes such as age or income they must be grouped into intervals

DEMOGRAPHIC KEY	AGE	GENDER	INCOME LEVEL
1	20-24	Male	<\$20,000
2	20-24	Male	\$20,000-\$24,999
3	20-24	Male	\$25,000-\$29,999
18	25-29	Male	\$20,000-\$24,999
19	25-29	Male	\$25,000-\$29,999

- Minidimensions should have less than **100000 distinct combinations** in order to **preserve high performance**

93

93

Demographic Minidimension

Customer Dimension				
CustomerKey	ID	FirstName	...	DemographicKey
1	121	James	...	1
2	45	Robert	...	3
3	327	John	...	1
4	921	Michael	...	18
5	431	William	...	1
6	233	Richard	...	2

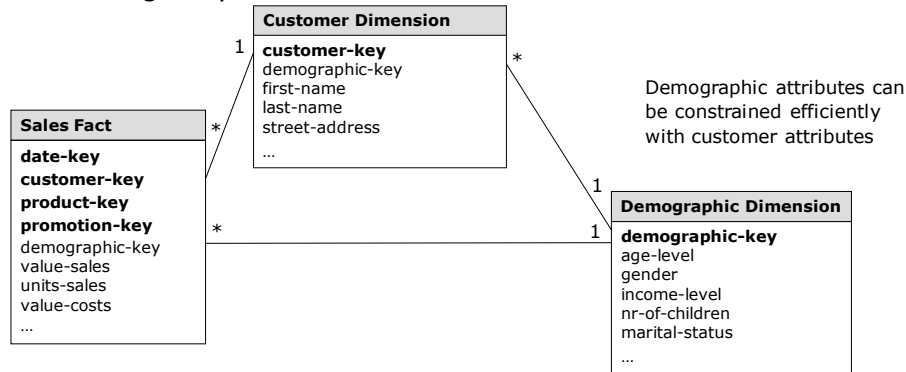
Demographic Dimension			
DemographicKey	Age	Gender	Income Level
1	20-24	Male	<20000\$
2	20-24	Male	20000\$-24999\$
3	20-24	Male	25000\$-29999\$
...
18	25-29	Male	20000\$-24999\$
19	25-29	Male	25000\$-29999\$

94

94

Demographic Minidimension

- Demographic minidimension **can be joined either directly to the fact table** and also to the **customer dimension**, by adding the foreign key to both tables



95

95

Rapidly Changing (Mini)Dimension

- Minidimensions drastically improve browsing and constraining performance
- Minidimensions are very useful because **the attributes isolated in the minidimensions are mainly the ones most used and/or that change more frequently** and whose distinct values must be tracked
- When a customer's demographic data changes, is only necessary to **insert a new record in the Customer dimension with a different demographic key** (SCD Type 2)

96

96

Rapidly Changing (Mini)Dimension

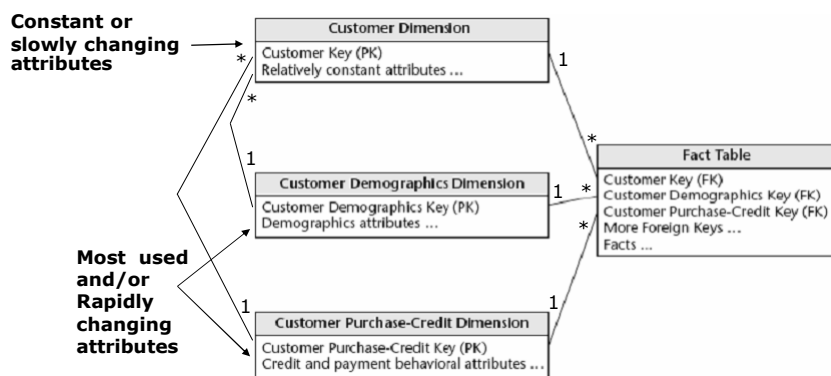
Customer Dimension						
CustomerKey	ID	FirstName	...	DemographicKey	StartDate	EndDate
1	121	James	...	1	17/03/2021	
2	45	Robert	...	3	17/03/2021	
3	327	John	...	1	17/03/2021	23/09/2023
4	921	Michael	...	18	17/03/2021	
...
10932	327	John	...	2	24/09/2023	

Demographic Dimension			
DemographicKey	Age	Gender	Income Level
1	20-24	Male	<20000\$
2	20-24	Male	20000\$-24999\$
3	20-24	Male	25000\$-29999\$
...
18	25-29	Male	20000\$-24999\$
19	25-29	Male	25000\$-29999\$

97

97

Large Changing Customer Dimension



Best approach for **efficiently tracking changes of key attributes in really huge dimensions** is **to break in one or more minidimensions** from the dimension table, each consisting of small clusters of attributes that have been managed to have a limited number of values (no more than 100000)

98

98

Determining Dimensions and Facts

99

Determining Dimensions and Facts

- Starting point for determining what data should be in a data mart
 - **Initial questions/analysis** the users want answered
 - **Operational databases**
- It is not possible to perfectly predict everything, that is why the **most granular/detailed model is always the best solution**
- Each question/analysis can be broken down into discrete items of business
 - Information the user wants to know – **facts**
 - Criteria used to access – **dimension attributes**
- Easy way to model the questions/analysis is through a **Matrix**
 - **Rows** are the **dimension attributes**
 - **Columns** are the **facts**
 - **Cells** contain the **number of the question/analysis**

100

100

Example of Determining Dimensions and Facts

Questions/Analysis to be answered:

1. What was the sales amount of health and beauty products in Europe to customers over age of 50 in each of the past three years ?
2. What is the name of salesperson who had the highest sales amount of each product in the first quarter of this year ?
3. How many European customers complaints did we receive on food products during the past year ? How has it changed from month to month this year ?
4. What are the name of stores that had the highest monthly quantity sales of sport clothing during the summer ?

101

101

Matrix

Dimensions/Facts	Sales Amount	Number of complaints	Sales quantity
product category	1	3	4
customer region	1	3	
customer age	1		
year	1;2	3	
salesperson name	2		
product	2		
quarter	2		
month		3	4
store name			4
season			4

102

102

Kimball Dimensional Modeling Steps

1. Business process

- Sales
- Complaints

2. Granularity

- Monthly sales by product by customer by store by salesperson
- Monthly complaints by product by customer

3. Dimensions

- Product
- Customer
- Date
- Salesperson
- Store

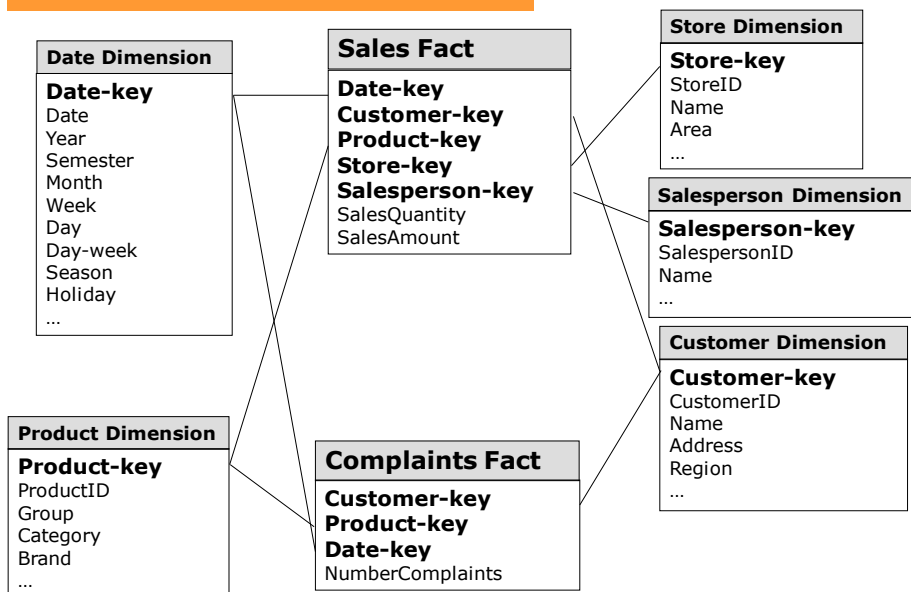
4. Facts

- Sales amount
- Sales quantity
- Number of complaints

103

103

Dimensional Model



104

104