

PA220: Database systems for data analytics

Data Warehouse Modelling

Contents

- Business event
- Dimensional modelling
- Star schema
- Data warehouse design
- Slowly changing dimension
- Date dimension
- Degenerate dimension

Transactional Database Design (OLTP)

- OLTP performance is about inserting and updating quickly
- Locking must be minimized
- Very small sets of data are retrieved in a query
- Data consistency is critical
- Laws of normalization

Reporting Database Design (Data Warehouse)

- Copy of OLTP
- Performance is about retrieving the data quickly
- Locking is not an issue
- Large sets of data are retrieved in a query
- Insert and Update speed is not important.

Business Process Measurements

- Want to store information about a business process.
 - -> Store “business process measurement events”
- Example: Retail sales
 - -> Could store information like date/time, product, store number, promotion, customer, clerk, sales dollars, sales units, ...
 - -> Implies a level of detail, or grain.
- Observe: These stored data have different flavors:
 - Some refer to other entities, e.g., to describe the context of the event (e.g., product, store, clerk)
 - -> dimensions
 - Some look more like “measurement values” (sales dollars, sales units)
 - -> facts or measures

Business Process Measurement Events

- A flat table view of the events could look like

State	City	Quarter	Sales Amount
California	Los Angeles	Q1/2013	910
California	Los Angeles	Q2/2013	930
California	Los Angeles	Q3/2013	925
California	Los Angeles	Q4/2013	940
California	San Francisco	Q1/2013	860
California	San Francisco	Q2/2013	885
California	San Francisco	Q3/2013	890
California	San Francisco	Q4/2013	910
⋮	⋮	⋮	⋮

Business Analysts

- Businesspeople are used to analyzing such data using pivot tables in spreadsheet software.

The image shows a spreadsheet with a pivot table and the PivotTable Builder dialog box. The pivot table is located in the range C4:F10 and displays sales data by city and quarter. The PivotTable Builder dialog box is open, showing the configuration for the pivot table.

Row Labels	Q1/2013	Q2/2013	Q3/2013	Q4/2013	Grand Total
Austin	510	495	535	505	2045
Dallas	595	610	615	605	2425
Houston	550	605	555	585	2295
Los Angeles	910	930	925	940	3705
San Francisco	860	885	890	910	3545
Grand Total	3425	3525	3520	3545	14015

The PivotTable Builder dialog box shows the following configuration:

- Field name: Search fields
- Field list: State (unchecked), City (checked), Quarter (checked), Sales Amount (checked)
- Drag fields between areas:

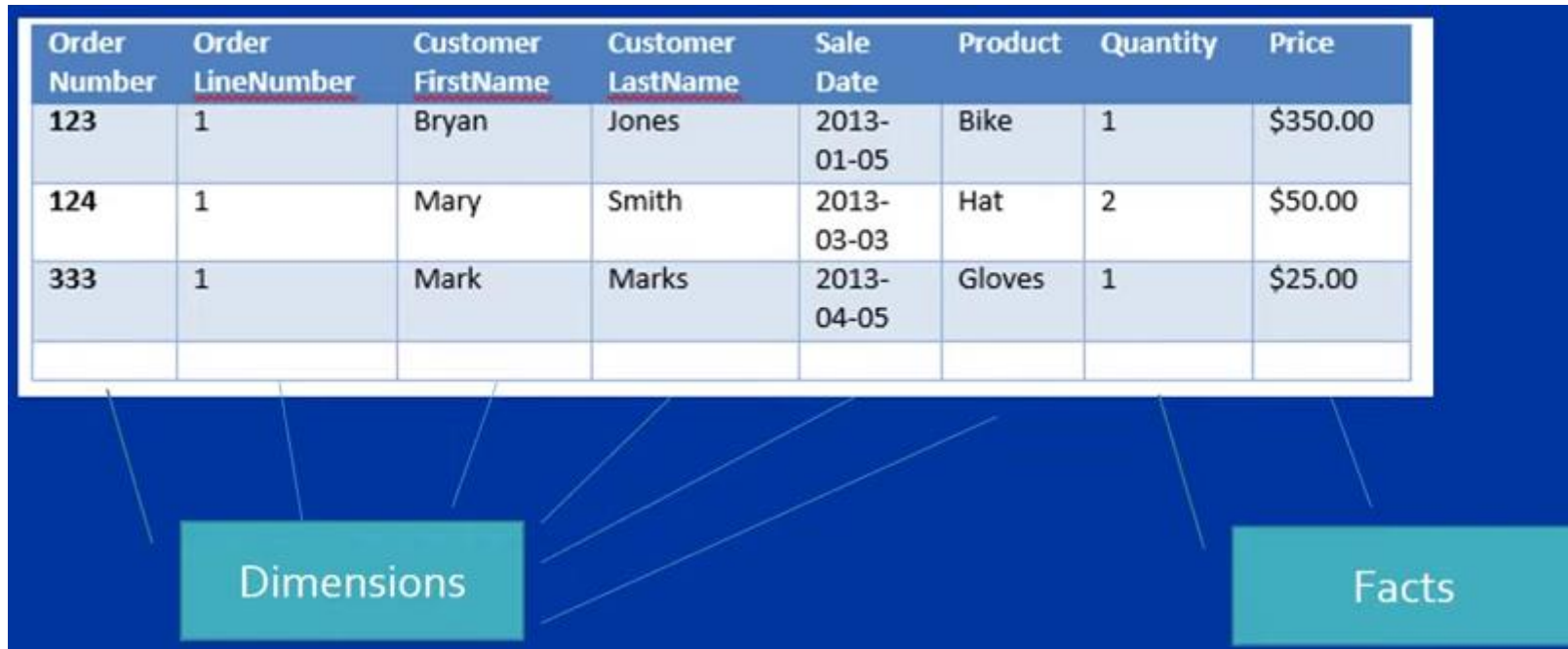
 - Report Filter: (empty)
 - Column Labels: Quarter
 - Row Labels: City
 - Values: Sum of Sa...

Dimensional Modeling

- Ralph Kimball: "Dimensional Modeling is a design technique for databases intended to support end-user queries in a data warehouse"
- Event table attributes:
 - use some as dimensions and some as measures to aggregate (facts)
- Descriptive data is separated from quantity
 - e.g., customer name is separated from the order amount.

Dimensional Modeling (2)

An analysis of flat table:




Dimensional Modeling (3)

- Data is denormalized as needed to support reporting.
- The resulting model reflects the kinds of questions the business wants to ask rather than the functions of the underlying operational system.
- Data maintenance performance is secondary.

What is a Dimension?

- Dimensions describe business events like the sale of a product.
- They are what users would want to **sort**, **group** and **filter**
 - e.g., dates, customer number, store number

An example of a dimension...



Store_key	Store_desc	City	State	Zip_code	Region	District
-----------	------------	------	-------	----------	--------	----------

A user might want to group by store state.

Dimension

- Typical criterion for grouping
- Many dimensions support some form of hierarchy
 - E.g., city -> region -> state -> country
- Sometimes: more than one natural hierarchy
 - E.g., date -> day -> month decade -> month -> quartal -> year
 - -> weekday
- Bottom level of hierarchy
 - the finest granularity
 - -> which leads to the definition of *atomic grain in the fact table*.
- Top level of hierarchy
 - typically, “ALL”

What is a fact?

- A fact is a measurable metric which is described by the dimensions.
 - e.g., the sale amount or order quantity
 - Sometimes it is simply called a measure.
- There are usually many more dimensions than facts.

An example of a fact table...

Sales_Fact	
<u>Time key</u>	
<u>Product key</u>	
<u>Store key</u>	
<u>Sales Person Key</u>	
Sale_amount	
Unit_price	
Discount	
Units	

Sales amount is a fact.
We would likely want to summarize it.

The diagram shows a table named 'Sales_Fact' with a grid icon in the top left corner. The table has two columns. The first column contains the following rows: 'Time key', 'Product key', 'Store key', 'Sales Person Key', 'Sale_amount', 'Unit_price', 'Discount', and 'Units'. The first four rows have a horizontal line under the text, indicating they are keys. A light blue callout box with a white border is positioned to the right of the table, containing the text 'Sales amount is a fact. We would likely want to summarize it.' A thin blue line points from the 'Sale_amount' row in the table to the callout box.

Facts

- Facts represent the subject of the desired analysis
 - The "important" in the business that should be analyzed
- A fact is most often identified via its dimension values
 - A fact is a non-empty cell (in the pivot table)
 - Some models give facts an explicit identity
- Generally, a fact should
 - be attached to exactly one value in each dimension;
 - only be attached to dimension values in the bottom levels
 - e.g., if the lowest time granularity is day, for each fact the exact day should be specified;
 - some models do not require this.

Facts (2)

- Six different types of facts are distinguished

- Transaction (Event) facts

- A fact for every business event (sale, order, booking)
 - Can optionally contain the precise timestamp as a degenerate dimension

- Snapshot facts

- A fact for every dimension combination at given time interval (period), which is the grain of it.
 - Typically, uniformly dense – zero/null-value in facts for periods of “no activity” → *Periodic snapshot fact*
 - Captures current state (inventory)

- Cumulative snapshot facts

- A fact for every dimension combination at given time interval
 - from the beginning to the end of a process (e.g., claim processing)
 - Captures cumulative status up to now, e.g., total sales to date
 - So, the fact row is updated

- “Fact-less” facts

- A fact per event (customer contact)
 - No numerical measures
 - An event happened for a dimension value combination

An example of a fact table...

The diagram shows a fact table named 'Sales_Fact' with a grid icon in the top left corner. The table is divided into two sections: dimensions (keys) and measures. The dimensions are 'Time key', 'Product key', 'Store key', and 'Sales Person Key', each with a horizontal line underneath. The measures are 'Sale_amount', 'Unit_price', 'Discount', and 'Units', listed below the dimensions. A blue arrow points from the text 'An example of a fact table...' to the 'Sales_Fact' table.

Sales_Fact	
<u>Time key</u>	
<u>Product key</u>	
<u>Store key</u>	
<u>Sales Person Key</u>	
Sale_amount	
Unit_price	
Discount	
Units	

Facts (3)

- Different types of facts are distinguished
 - Aggregated facts
 - simple numeric rollups of atomic fact table
 - just to accelerate queries, so “pre-aggregates” not “source data”
 - Consolidated facts
 - facts combined from multiple processes if they can be expressed at the same grain.
 - e.g., sales actuals combined with sales forecasts
- **Every type of facts answers different questions**
 - Often transaction facts and snapshot facts exist

Fact Granularity (Grain)

- **Granularity** of facts is important
 - What does a single fact mean?
 - Determines the level of detail
 - *Given by the combination of bottom levels*
 - e.g., "total sales per store per day per product"
 - Important for scalability of the number of facts
- Often the granularity is a single business transaction
 - Example: sale
 - Sometimes the data is aggregated (total sales per store per day per product)
 - Aggregation might be necessary for scalability
- Generally, transaction detail can be handled
 - Except perhaps huge clickstreams, etc.

Measures

- Measures represent the fact value (property) that users want to study and analyze
 - e.g., the sales price
- A measure has two components
 - Numerical value: (sales price)
 - Aggregation formula: (SUM)
 - used for aggregating/combining several measure values into one
- Additivity is an important property for measures
 - Single fact table rows are (almost) never retrieved, but aggregations over millions of fact rows.
- Measure value determined by the combination of dimension values
 - Measure value is meaningful for all aggregation levels

Measures (2)

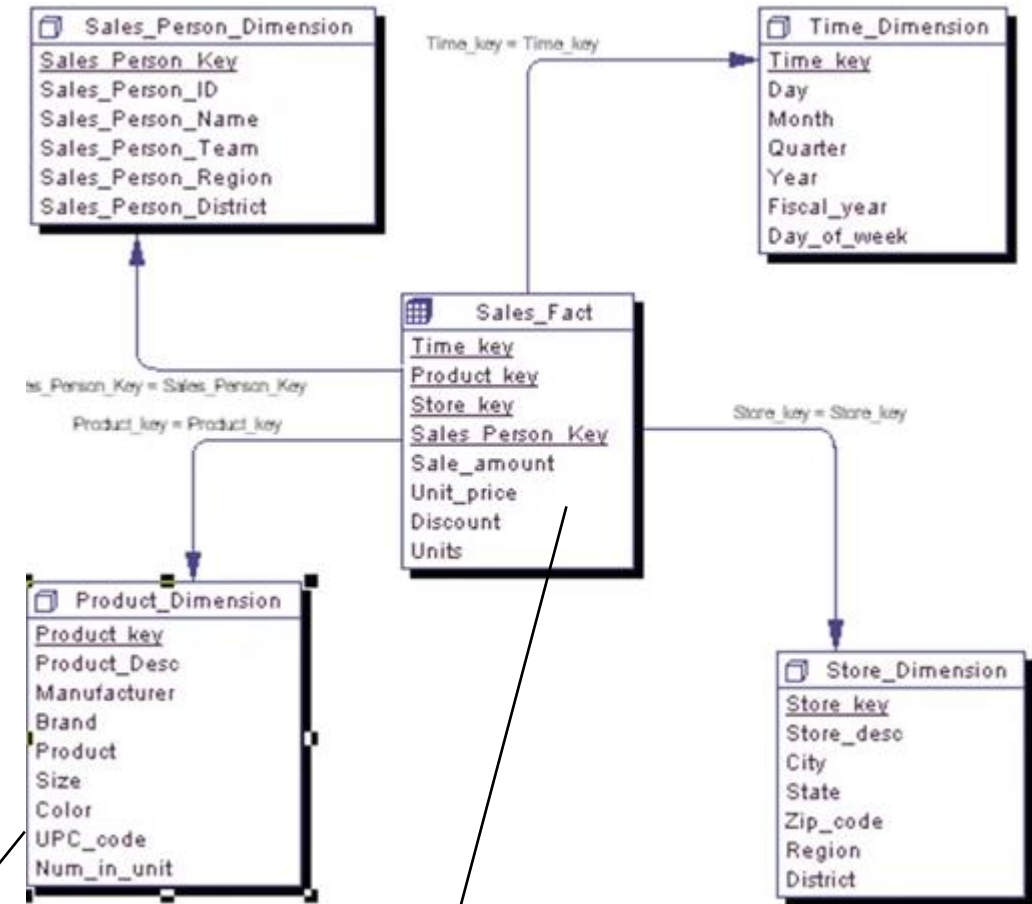
- Three types of measures are distinguished
 - (Fully-)Additive measures
 - Can be aggregated over all dimensions using SUM
 - e.g., sales price, gross profit computed from sales, and cost
 - Often occur in **transaction (event) facts**
 - Semi-additive measures
 - Cannot be aggregated over some dimensions – typically time
 - e.g., banking: account balance is additive across all dimensions but time
 - Often occur in **snapshot facts**
 - Non-additive measures
 - Cannot be aggregated over any dimensions
 - e.g., unit cost cannot be added, ratios
 - Occur in all types of facts

Measures (3)

- There are two more types of measures:
- Aggregated measure
 - Pre-computed totals or averages over certain dimensions
 - E.g., monthly sales totals or average daily traffic.
 - Corresponding to Aggregated and/or Consolidates fact table types
- Derived measure
 - Calculated from other measures, often using formulas
 - E.g., gross profit can be derived from revenue and cost measures.
 - May not be physically stored in the fact table

Star Schema

- Rather than a flat table, use a star schema for dimensional modelling in a relational database.
- What types of queries are supported?
 - How will “slice and dice” queries look like on such a schema?

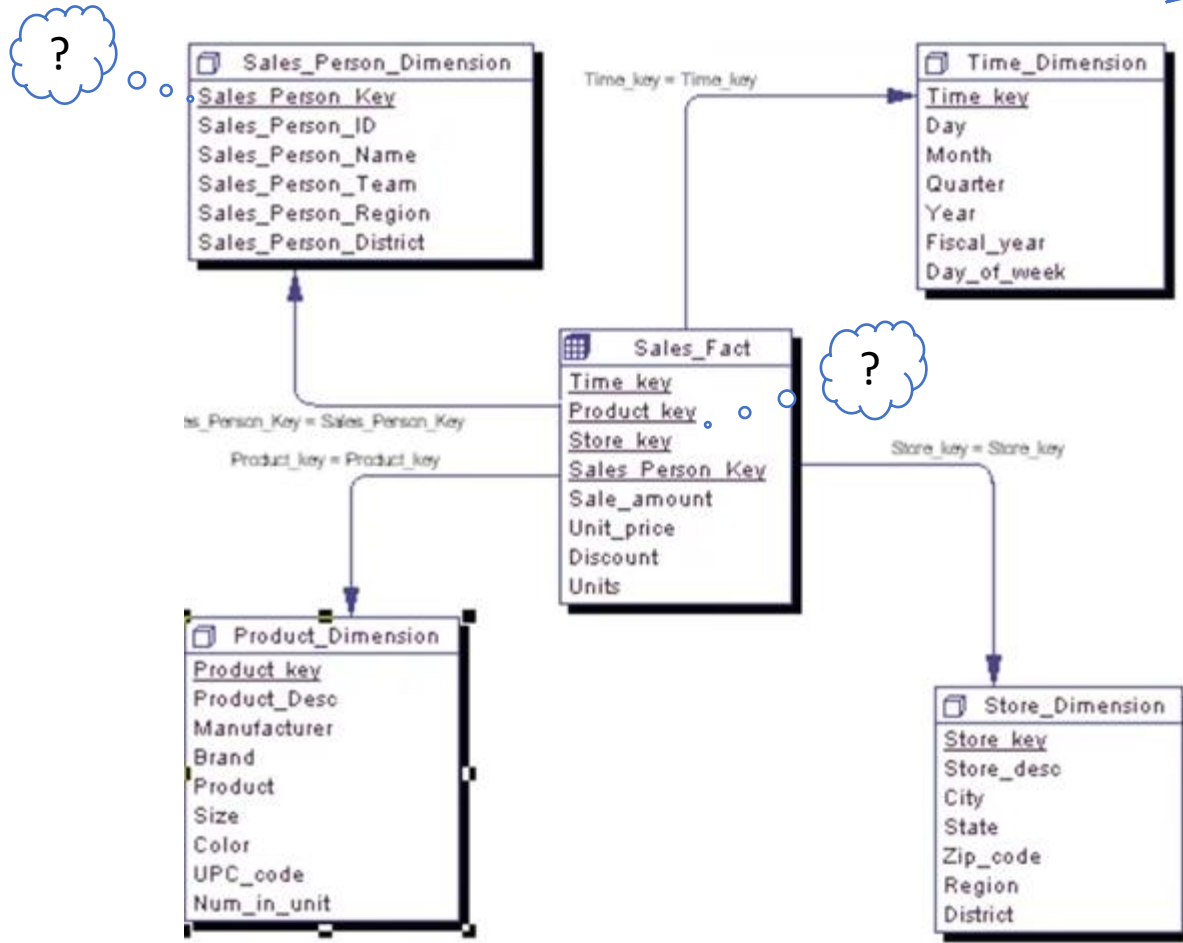


A dimension can contain many attributes

A fact table has facts (measures) and a key to each related dimension.

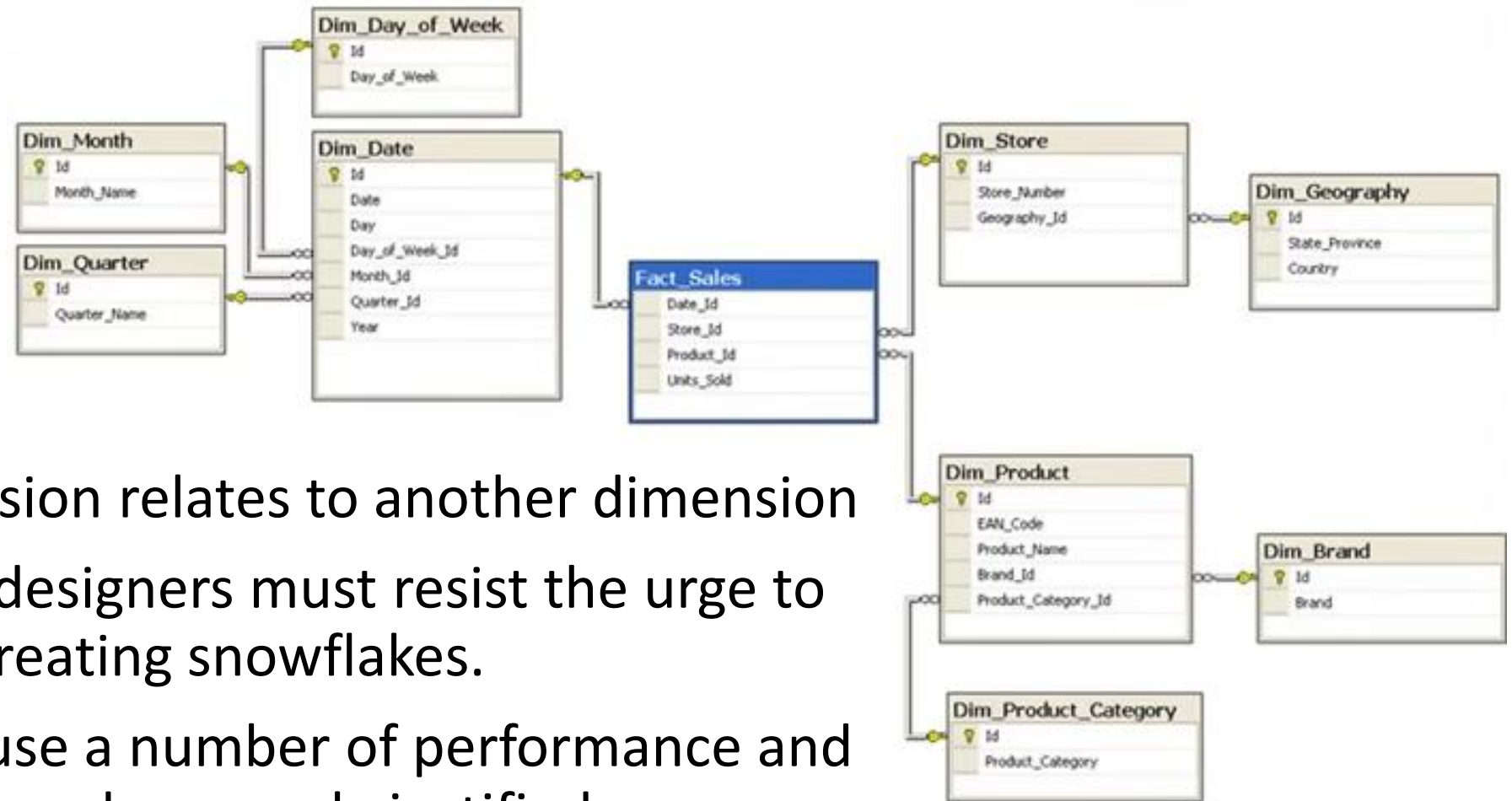
Warning! This is not a relational design so be careful if you are an OLTP developer.

Star Schema – Key Points



- Dimensions
 - relate directly to the fact table only.
 - are denormalized
 - i.e., Salesperson Region does not have a related region lookup table
 - as an OLTP design would likely have.
- Dimension keys are **not keys** from the source systems
 - they are **surrogate keys**
 - which are generated by DW load process
- Grain of the facts
 - defined by the dimension attributes
 - i.e., how detailed the measures are.
 - **primary key** is the combination of dim. FKs

The Snowflake Schema



- When a dimension relates to another dimension
- Beware! OLTP designers must resist the urge to normalize by creating snowflakes.
- Snowflakes cause a number of performance and usability issues and are rarely justified.

The Snowflake Schema

- + Hierarchies are made explicit/visible
- + Very flexible
- + Dimension tables use less space
 - However, this is a minor saving
 - Disk space of dimensions is typically less than 5 percent of disk for DW
- - Harder to use due to many joins
- - Worse performance
 - e.g., efficient bitmap indexes are not applicable

Some Key Terms

- Surrogate Keys
 - artificially created keys (usually integers) used only by the data warehouse to uniquely identify a row in a dimension table.
- Grain
 - level of detail a fact row represents. For example, sale amount of a single item at a given date/time by salesperson A in the Boston store.
 - Must be also consistent with the level of detail of dimensions of the fact table.
- Atomic grain
- Conformed Dimension
 - Different source systems (CRM versus Sales) often have differences in the list of dimension values they support. The CRM system may not have enclosed branches, but the sales system does.
 - A consolidated list of dimension values that supports all the source systems values is called a conformed dimension.
 - Conformed dimensions are critical for a successful data warehouse.

Surrogate Keys

- Surrogate keys are integers
 - assigned sequentially to records in a dimension table, e.g., 1, 2, 3, ...
- Should be used instead of natural operational production codes
 - Surrogate key will be the primary key of a dimension table.
 - Original (natural) primary key is typically added as a regular attribute in a dimensional table.

Product
id_product (PK)
name
id_brand (FK)
id_category (FK)

Operational database



Dim_product
id_product (PK)
name
brand
category

Data warehouse



Dim_product
id (PK)
id_product
name
brand
category

Importance of surrogate keys

- Advantages

- Make the DW independent from unexpected operational key changes
 - e.g., operational DB may re-use of old (unused) operational keys after some time
- Avoid key overlap problem when consolidating data
 - e.g., different source systems may have the same entity under different IDs
- Dimension keys should not contain “intelligence”
 - some semantics known in operational stores, e.g., “PA220” as course code
 - any such should be converted in dimension attributes
- Handle unknown (NULL) values
 - e.g., missing date in invoices -> NULL in FK in the fact table is not allowed!
- Better performance, since surrogate keys are tight and efficient
 - Results in smaller fact tables
 - 1 byte in a 1 billion-row fact table translates into 1 GB of disk space
 - e.g., small integer (2-, 4-byte) vs. long alpha-numeric code (PA220, BKM_DATS)

- Disadvantages

- Needs to implemented, but it is reusable for any other dimension
- More space needed in dimensional tables

More thoughts on surrogate keys

- Required to implement a history in slowly changing dimensions.
- Avoids conflicts among backend application keys.
- Isolates the data warehouse from backend application changes.
- Different backend applications may use different columns as the dimension key.

Steps of Dimensional Modelling

1. Choose the business process
2. Declare the grain
3. Identify the dimensions
4. Identify the facts

DM Steps: 1. Choose the business process

The basics in the design build on the actual business process which the data warehouse should cover. Therefore, the first step in the model is to **describe the business process** which the model builds on.

This could for instance be a sales situation in a retail store. To describe the business process, one can choose to do this in plain text or use basic Business Process Modeling Notation (**BPMN**) or other design guides like the Unified Modeling Language (**UML**).

DM Steps: 2. Declare the grain

The grain of the model is the exact description of what the dimensional model should be focusing on.

This could for instance be "An individual line item on a customer slip from a retail store".

To clarify what the grain means, you should pick the central process and describe it with one sentence. Furthermore, the grain (sentence) is what you are going to build your dimensions and fact table from.

You might find it necessary to go back to this step to alter the grain due to new information gained on what your model is supposed to be able to deliver.

DM Steps: 3. Identify the dimensions

The dimensions must be defined within the grain from the second step of this 4-step process.

Typically, dimensions are nouns like date, store, inventory etc. These dimensions are where all the data is stored. For example, the date dimension could contain data such as year, month and weekday.

Dimensions are the foundation of the fact table and is where the data for the fact table is collected.

Mary Jones buys 1 book for \$22.50 entitled "Agile Data Warehouse Design" on December 2, 2013 at 3:12 PM via Amazon.com using her Visa card to be delivered on December 10, 2013 by UPS.

DM Steps: 4. Identify the facts

After defining the dimensions, the next step in the process is to make values for the fact table.

This step is to identify the **numeric facts** that will populate each fact table row. This step is closely related to the business users of the system, since this is where they get access to data stored in the data warehouse.

Therefore, most of the fact table rows are numerical, additive figures such as quantity or cost per unit, etc.

Mary Jones buys 1 book for \$22.50 entitled "Agile Data Warehouse Design" on December 2, 2013 at 3:12 PM via Amazon.com using her Visa card to be delivered on December 10, 2013 by UPS.

The Seven W's of DW Design

- One instance of the event:
 - Mary Jones buys 1 book for \$22.50 entitled "Agile Data Warehouse Design" on December 2, 2013 at 3:12 PM via Amazon.com using her Visa card to be delivered on December 10, 2013 by UPS.
- The W's:
 - How
 - What
 - When
 - Where
 - Who
 - How many/much
 - Why

The Seven W's of DW Design

The W's:

How

What

When

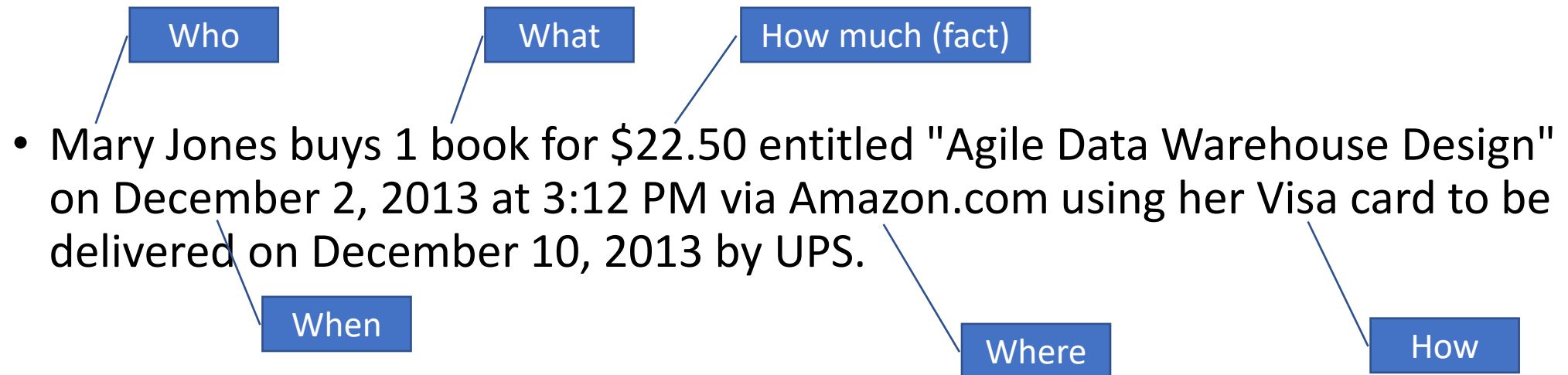
Where

Who

How many/much

Why

- One instance of the event:



The Seven W's of DW Design

- Intuitive and natural for business users
- Efficient way to get the required details
- Provides a jumping-off point to get other information
 - e.g., “Mary ordered via the internet. Are there other outlets for buying products?” or “Mary is an individual, do you have groups or corporate customers?”
- **Helps you focus on a single process at a time.**

Data Warehouse Bus Matrix

- Finding common dimensions

BUSINESS PROCESSES	COMMON DIMENSIONS							
	Date	Product	Store	Promotion	Warehouse	Vendor	Contract	Shipper
Retail Sales	X	X	X	X				
Retail Inventory	X	X	X					
Retail Deliveries	X	X	X					
Warehouse Inventory	X	X			X	X		
Warehouse Deliveries	X	X			X	X		
Purchase Orders	X	X			X	X	X	X

Figure 3.8 Sample data warehouse bus matrix.

- Create one dimension per column
- Create one fact table per row
- Alternatively called *Enterprise Bus Matrix*

Data Warehouse Bus Matrix

- Conformed dimensions
 - dimension referenceable from different business processes
 - good DW design calls for just conformed dimensions

BUSINESS PROCESSES	COMMON DIMENSIONS							
	<i>Date</i>	<i>Product</i>	<i>Store</i>	<i>Promotion</i>	<i>Warehouse</i>	<i>Vendor</i>	<i>Contract</i>	<i>Shipper</i>
Retail Sales	X	X	X	X				
Retail Inventory	X	X	X					
Retail Deliveries	X	X	X					
Warehouse Inventory	X	X			X	X		
Warehouse Deliveries	X	X			X	X		
Purchase Orders	X	X			X	X	X	X

Figure 3.8 Sample data warehouse bus matrix.

Slowly Changing Dimension (SCD)

- Dimension values can change
- How the change is handled depends on the value the business places on knowing the historical values of a dimension.
- Example:
 - The New York store is reassigned from the Northeast Region to the Middle Region.
 - Does management want to be able to see changes in sales as the Northeast Region before the change and the Middle Region after the change?

SCD Type 1: Overwrite

- Simply overwrite the existing dimension data with the new information
 - Adv: Easy to implement
 - Disadv: Lose ability to see how data looked in past -> no history of values
- Example:
 - After the change, all sales in history of the NY store will be counted to the Middle Region.

SCD Type 2: Add New Row

- Keep all historical values – take snapshots / make versioning
 - Adv: Better ability to report accurately historically.
 - Disadv: Most complex to implement.
- Example:
 - Sales in New York store made prior to the change will be reported in the Northeast Region
 - Any sales after that will be reported in the Middle Region.

id	store_name	store_city	store_region	valid_from	valid_to	current_row
123	SkirtJoyce	New York	NorthEast	2010-01-14	2020-09-21	expired
124	SkirtJoyce	New York	Middle	2020-09-22	NULL	current

New surrogate key

Are NULLs allowed?

SCD Type 2: Implications to data

- Updating a row in the dimension table implies
 - a new surrogate key value, which implies
 - updating the fact table (change all respective recs to the new value) → possibly slow; or
 - **querying for all data – use multiple keys of the same store.**
 - But can I find the previous versions from the current one?
 - We must add a new attribute for making the relationship, e.g., store_id, or the original key.
 - leaving the same value → dropping the primary key constraint
 - no changes to the fact table, but the foreign key must be dropped too.
 - not common

SCD Type 3: Add New Attribute

- Keep the prior value and the new value
 - Adv: Easier to implement than SCD type 2 while still providing some support for historical reporting
 - Disadv: Historical reporting is limited.
- Example:
 - Sales in New York store made prior to the change will be reported in the Northeast Region but any sales after that will be reported in the Middle Region.
 - However, if the store was moved to the Southeast Region subsequently, you would lose that it was ever in the Northeast Region.

id	store_name	store_city	store_region	prev_store_region	valid_from
123	SkirtJoyce	New York	Middle	NorthEast	2020-09-22

SCD Type 4: Add Mini-Dimension

- For rapidly changing attributes of a dimension
- Introduces a new mini-dimension over such attributes
 - Fact table is extended by the new dimension

Store dimension

id	store_name	store_city
123	SkirtJoyce	New York

Region dimension

id	store_region	valid_from	valid_to
1	NorthEast	2010-01-14	2020-09-21
2	Middle	2020-09-22	9999-12-31

Attributes valid_from/to were added here just to keep the information when the change happened. It is not required by design.

SCD Type 5: Mini-dim & Type 1 Outrigger

- Preserves historical attribute value and historical facts as of current attribute values.

Store dimension	id	store_name	store_city	cur_region
	123	SkirtJoyce	New York	2

Region dimension	id	store_region	valid_from	valid_to
	1	NorthEast	2010-01-14	2020-09-21
	2	Middle	2020-09-22	9999-12-31

SCD Type 6

- Preserves history as Type 5
 - current value is added on top.
 - on change, all rows of the “durable” key are systematically updated.
- Mix of types 1, 2 and 3

id	store_id	store_name	store_city	cur_store_region	prev_store_region	valid_from	valid_to	current_row
123	NY1	SkirtJoyce	New York	MiddleEast	NorthEast	2010-01-14	2020-09-21	expired
124	NY1	SkirtJoyce	New York	MiddleEast	Middle	2020-09-22	2021-07-31	expired
348	NY1	SkirtJoyce	New York	MiddleEast	MiddleEast	2021-08-01	9999-12-31	current

- Makes roll-ups easier for current value
 - cur_store_region is available in all rows

SCD Type 0: Retain Original

- Passive method
- when source value changes, we do not do the update in DW.
 - Ignore changes ((-:
- E.g., the attribute is obsolete – fax number

SCD Types – Summary

- SCD 0 - Passive method (No Change)
- SCD 1 - Overwriting the old value (Latest Record Only)
- SCD 2 - Creating a new additional record (Maintains History)
- SCD 3 - Creating an additional Column (Rarely Used)
- SCD 4 - Using history table, for rapidly changing dimensions
- SCD 5 - History table and “nested” dimension
- SCD 6 - The hybrid approach of overwriting, adding attrs and history
- SCD 7 - Dual type 1 and type 2 dimensions
 - places both the durable and surrogate key to the fact table

The Date Dimension

- Typically uses the date value in integer form, e.g., 20130101 as its key.
- Used to provide descriptive date information, i.e., month name, quartal, day of week, etc.
- Often has multiple keys in the fact table that point to it. There are called the role-playing dimensions.
 - Can be substituted by view to differentiate the references semantically.
- Load with all past and future dates possible from the data.
 - aka created in advance
- SSAS has a wizard to generate this dimension data.

The Date (/Time) Dimension

- “Meaningful” values are important, e.g., for report generation
 - Holiday/Nonholiday vs. Yes/no
 - Some other events,
- Time-of-day is a **separate** dimension
- Date dimension vs. SQL date type
 - Many date attributes are not supported in SQL, e.g., fiscal month
 - Business user is not versed in SQL
 - Date dimension is relatively small
 - 10 years = 3,650 rows

Date Dimension

DateKey

Date

Full Date Description

Day of Week

Day Number in Epoch

Week Number in Epoch

Day Number in Calendar Month

Day Number in Calendar Year

Last Day in Week Indicator

Last Day in Month Indicator

Calendar Week Ending Date

Calendar quarter

Calendar Year-Quarter

Calendar Half Year

Calendar Year

Fiscal Week

Fiscal Month

Fiscal Quarter

Fiscal Half Year

Fiscal Year

Holiday Indicator

Weekday Indicator

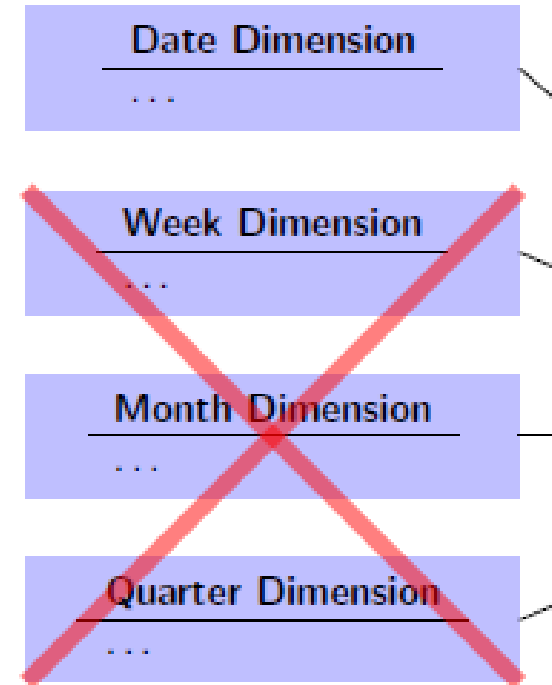
Selling Season

Major Event

...

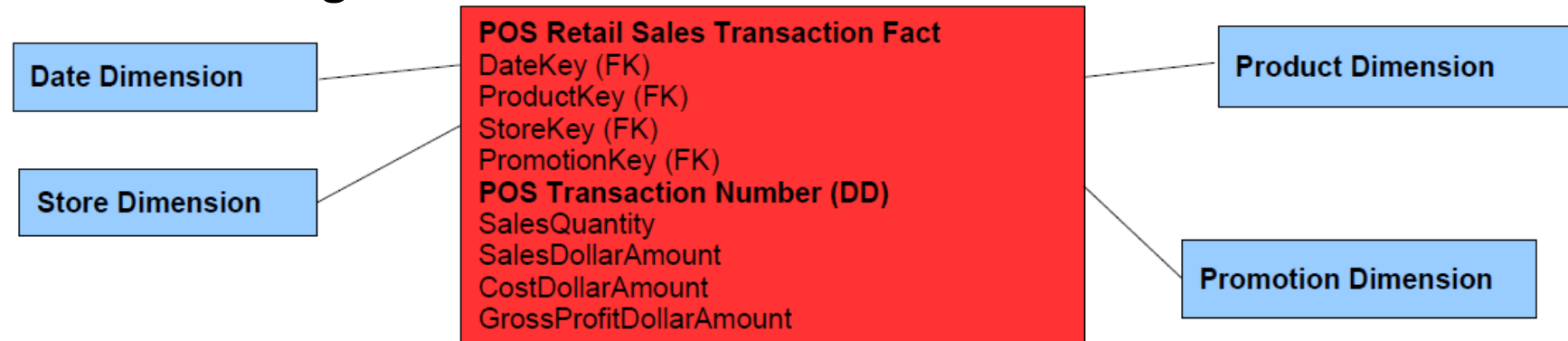
The Date (/Time) Dimension

- Impractical division
- Dimensions provide detailed information for the analysis
 - A sign that dimensions are not independent and hence should be combined
- Too many dimensions is bad
 - Significantly increases space requirements of fact table
 - Size of dimension table is not a problem



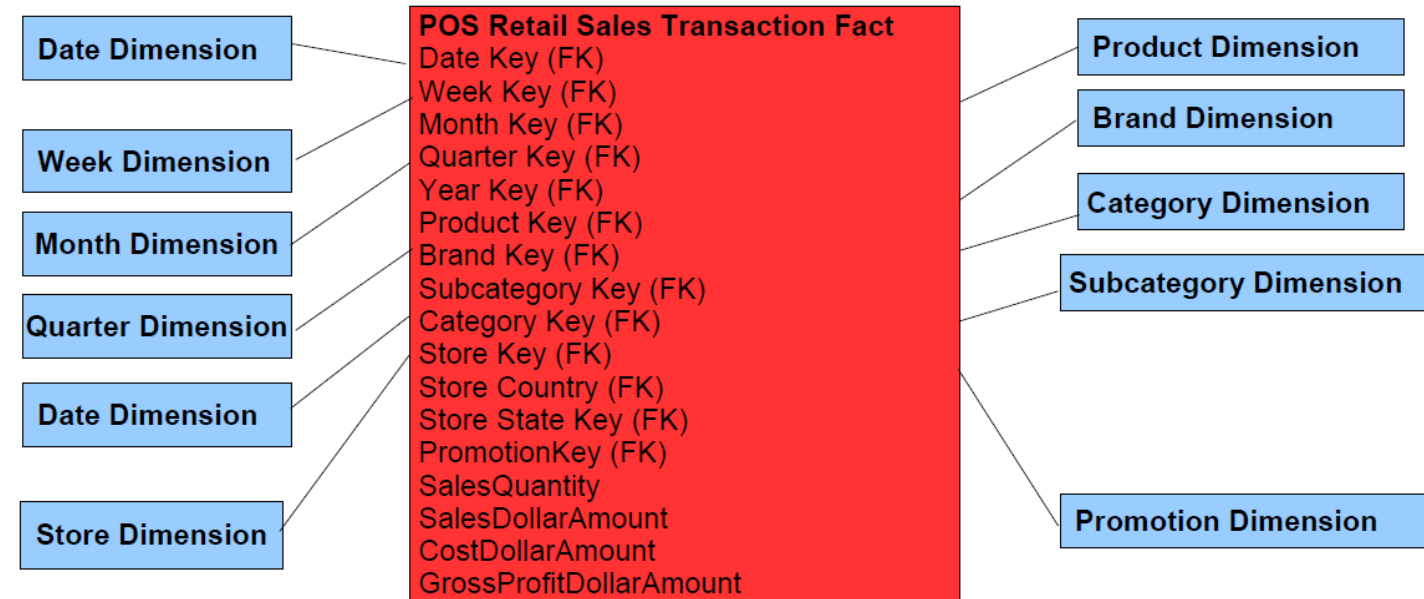
Degenerate Dimensions

- “empty”, i.e., dimension keys without dimension table
- Examples are operational control numbers
 - e.g., order #, invoice #, POS transaction #, etc.
 - Contain important information
 - Still useful to serve as part of primary key in fact table or for grouping
 - e.g., grouping by POS transaction number to retrieve all products purchased in a single transaction



Amount of Dimensions

- Typically, 4-15 per fact table
- Too many dimensions is bad
 - A sign that dimensions are not independent, and hence should be combined
 - Significantly increases space requirements of fact table



Redundancy in DW

- Only very little redundancy in fact tables
 - The same fact data (generally) only stored in one fact table
- Redundancy is mostly in dimension tables
 - Star dimension tables have redundant entries for the higher levels
- Redundancy problems?
 - Inconsistent data – the central load process helps with this
 - Update time – the DW is optimized for querying, not updates
 - Space use – dimension tables typically take up less than 5% of DW
- So: controlled redundancy is good
 - Up to a certain limit

Summary

- Dimensional modelling – four steps
- Key terms: surrogate keys, grain, conformed dim., bus matrix
- Star schema vs. snowflake schema
- Slowly changing dimensions
- Date dimension
 - can become a role-playing dimension, when referenced multiple times (issue date, settlement date, due date, ...)