Advanced Data Management Technologies

Unit 6 — Case Studies

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE

Acknowledgements: I am indebted to Michael Böhlen and Stefano Rizzi for providing me their slides, upon which these lecture notes are based.

Outline

1. The Grocery Store Example
2. More about Multidimensional Modeling
3. Inventory Management Example

Outline

1. The Grocery Store Example
2. More about Multidimensional Modeling
3. Inventory Management Example

The Grocery Store Example/1

- A grocery chain with 500 stores spread over a five-state area.
  - Each of the stores is a typical modern supermarket with a full complement of departments including grocery, frozen foods, dairy, meat, bakery, hard goods, liquor, and drugs.
  - Each store has roughly 60.000 individual products on its shelves.
- The individual products are called stock keeping units (SKUs).
- About 40.000 of the SKU come from outside manufacturers and have bar codes imprinted on the product package.
  - These bar codes are called universal product codes (UPCs).
  - UPCs are at the same grain as individual SKUs.
  - Each different package variation of a product has a separate UPC and hence is a separate SKU.
- The remaining 20.000 SKUs come from departments like meat or bakery departments and do not have nationally recognized UPC codes.
  - The grocery store assigns SKU numbers to these products by sticking scanner labels on the items.
  - Although the bar codes are not UPCs they are certainly SKU numbers.

The Grocery Store Example/2

- Data is collected at several places in a grocery store.
  - Some of the most useful data is collected at the cash registers as customers purchase products.
  - Our modern grocery store scans the bar codes directly into the point-of-sale (POS) system.
  - The POS system is at the front door of the grocery store where customer takeaway is measured.
  - The back door, where vendors make deliveries, is another interesting data-collection point.

The Grocery Store Example/3

- At the grocery store, management is concerned with the logistics of ordering, stocking the shelves, and selling the products **while maximizing the profit** at each store.
- The profit ultimately comes from
  - charging as much as possible for each product,
  - lowering costs for product acquisition and overhead, and
  - at the same time attracting as many customers as possible.
- The most significant decisions have to do with pricing and promotions.
  - Both store management and headquarters marketing spend a great deal of time tinkering with pricing and running promotions.
  - Promotions in a grocery store include temporary price reductions, ads in newspapers and newspaper inserts, displays in the grocery store, and coupons.

Simplified DM Design Process (Kimball and Ross)

- A somehow simplified DM design process consists of the following 4 steps:
  1. Choose the business process(es) to model
     - e.g., Sales,
  2. Choose the granularity of the business process
     - e.g., Items by Store by Promotion by Day.
     - Low granularity is needed.
     - Are individual transactions necessary/feasible?
  3. Choose the dimensions
     - Time, Store, ...
  4. Choose the measures
     - Dollar_sales, unit_sales, dollar_cost, customer_count

Step 1: Choose the Business Process

- A business process is an activity in the organization that typically is supported by a source data management system
  - raw material purchasing, orders, shipments, invoicing, inventory, bank transfers, patient transfers, ...
- Business processes are not necessarily limited to a single department
  - e.g., sales and marketing departments might be interested in the orders
- Focusing on the business process rather than the department avoids duplication of work and keeps data more consistent
- The first dimensional model built should be the one with the most impact
- It should answer the most pressing business questions and be readily accessible for data extraction

Step 1: Choose the Business Process – Example

- Management wants to better understand customer purchases as captured by the POS system.
- Business process: POS retail sales
- Allows us to analyze:
  - What products are selling?
  - In which stores?
  - On what days?
  - Under what promotional conditions?
  - etc.

Step 2: Choose the Grain of the Business Process

- Preferably develop dimensional models for the most atomic information captured by a business process
  - Not because queries report individual rows, but queries need to cut through the details in very precise ways
- The more detailed/atomic data is, the more things we know
- Atomic data provides maximum analytic flexibility
- Can be constrained and rolled up in every possible way
- It is always possible to declare higher-level grains by aggregation of atomic data; the opposite is not true
- Less granular model is vulnerable to unexpected requests for more details
- Example grain declarations
  - Individual line item on a customer's sales ticket as measured by a scanner
  - An individual boarding pass of a flight
  - A monthly snapshot for each bank account

Step 2: Choose the Grain of the Business Process – Example

- Individual line item on a POS transaction is the most detailed data, and we choose this as grain ($\rightarrow$ event fact)
- Allows a very detailed analysis of sales
  - Difference in sales on Monday vs. Sunday
  - Is it worthwhile to stock so many individual sizes of certain brands?
  - How many shoppers took advantage of the 50-cents-off promotion on shampoo?
  - Impact in terms of increasing sales when a competitive diet soda product was promoted
  - etc.
- Note that none of these queries calls for data from a specific transaction, but require detailed ways to slice data
  - Could not be answered if only aggregated values would be stored, e.g., daily summaries

Step 3: Choose the Dimensions

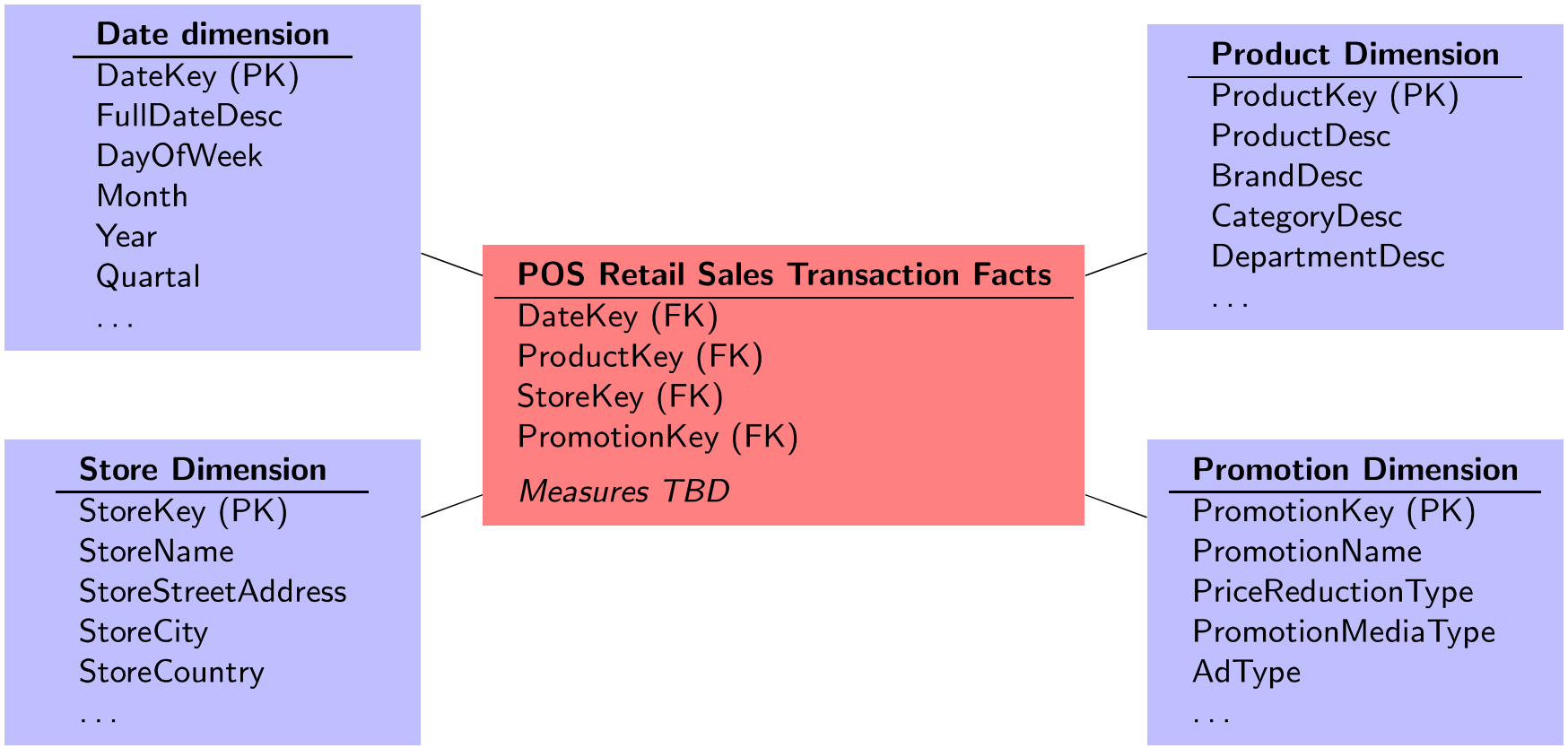
- Dimensions can be derived by answering the question *"How do business people describe the data resulting from the business processes?"*
- "Decorate" fact tables with dimensions representing all possible descriptions of the facts/measures
- A clear grain statement helps ~~to~~ identify the dimensions
- Sometimes a revision of step 2 is required

Step 3: Choose the Dimensions – Example/1

- The Date dimension
  - Explicit date dimension is needed (events, holidays, . . .)
- The Product dimension
  - Hierarchy allows drill-down/roll-up through category, brand, department, etc.
  - Many descriptive attributes (often more than 50)
- The Store dimension
  - Primary geographic dimension to specify location of the store
  - Many descriptive attributes
- The Promotion dimension
  - Used to see if promotions work and are profitable
  - Ads, price reductions, end-of-sale displays, coupons
    - Highly correlated (only 5000 combinations)

Step 3: Choose the Dimensions – Example/2

- Preliminary version of grocery store schema



Step 4: Choose the Measures

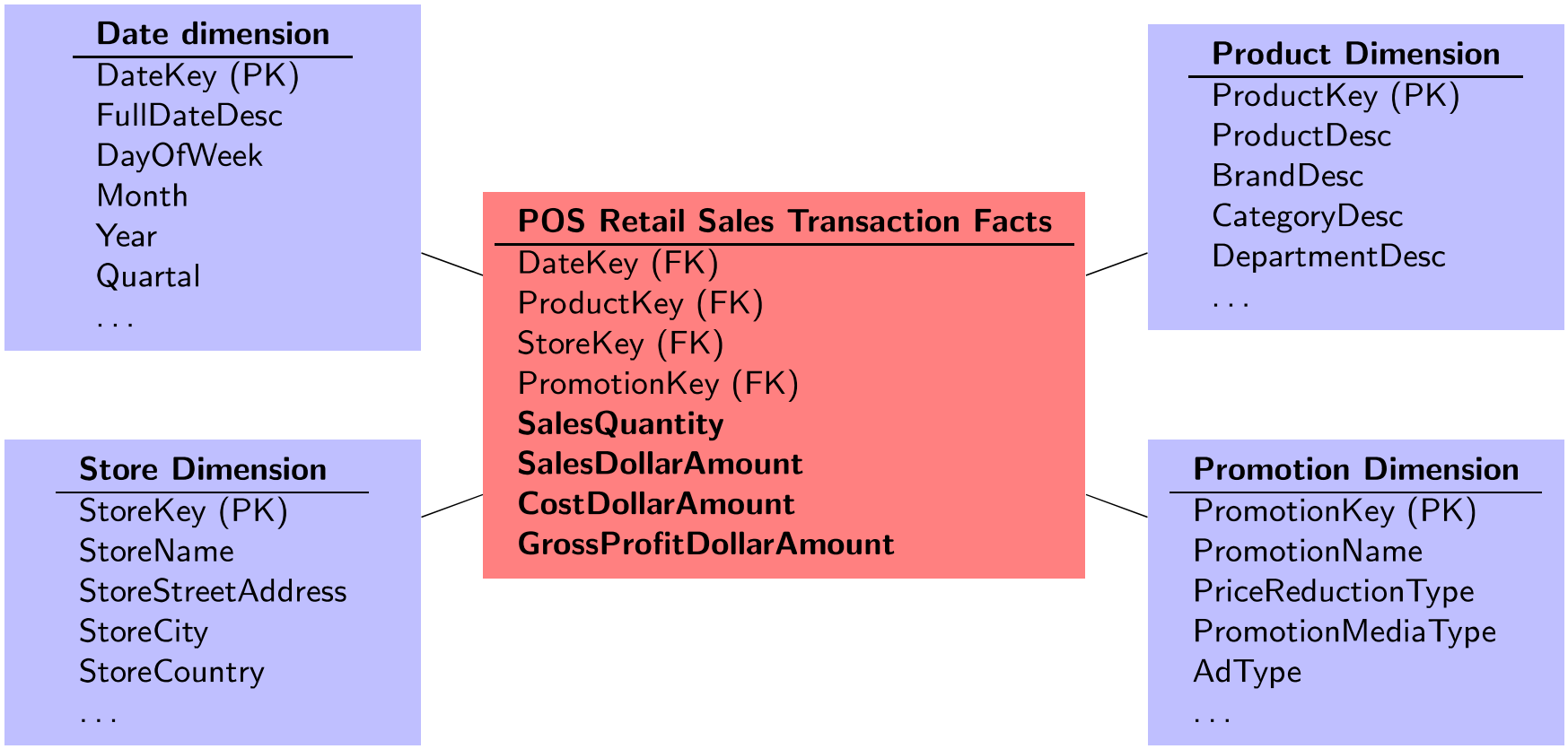
- Identify the business performance measures for the selected processes
- Determined by answering *"What are we measuring?"*
- Measures are determined by the grain declaration
- Revision of the grain might be required

Step 4: Choose the Measures – Example

- Sales quantity
- Sales dollar amount
- Cost dollar amount
- Gross profit
  - Equals to sales dollar − cost dollar
  - Explicit storage avoids user errors

- All measures are additive
- Facts are classified as event facts since each POS transaction is stored (most detailed level)

MD Schema of the Grocery Store Example



Outline

1 The Grocery Store Example

2 More about Multidimensional Modeling

3 Inventory Management Example

Database Sizing

- Time dimension: 2 years = 730 days
- Store dimension: 300 stores reporting each day
- Product dimension: 30,000 products, only 3,000 sell per day
- Promotion dimension: 5,000 combinations, but a product only appears in one combination per day
- Number of fact records: $730 \times 300 \times 3,000 \times 1 = 657,000,000$
- Number of fields: 4 key + 4 measures = 8 fields
- Total DB size: 657,000,000 × 8 fields × 4 bytes = 21 GB
- Small database by today's standards!

Date Dimension

- Date dimension is present in all DWs
- Can be created in advance
- "Meaningful" values are important for report generation, etc.
  - e.g., Holiday/Nonholiday vs. Yes/No
- Time-of-day a separate dimension
  - Separation keeps both dimensions small
- 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 (PK)
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
...

Instance of Date Dimension

Date Dimension Table

DK	Date	FullDateDescription	DayOfWeek	DayNum	HolidayInd	WeekdayInd	...
1	29.09.2013	September 29, 2013	Sunday	29	Nonholiday	Nonweekday	
2	30.09.2013	September 30, 2013	Monday	30	Nonholiday	Weekday	
3	01.10.2013	October 1, 2013	Tuesday	1	Nonholiday	Weekday	
4	02.10.2013	October 2, 2013	Wednesday	2	Nonholiday	Weekday	

Product Dimension

- Description of the products
- >50 attributes is typical for Product dimension
- Concept hierarchy
  - SKU → Brand → Category → Department
  - Many repetitions, but space of dimensions is not critical

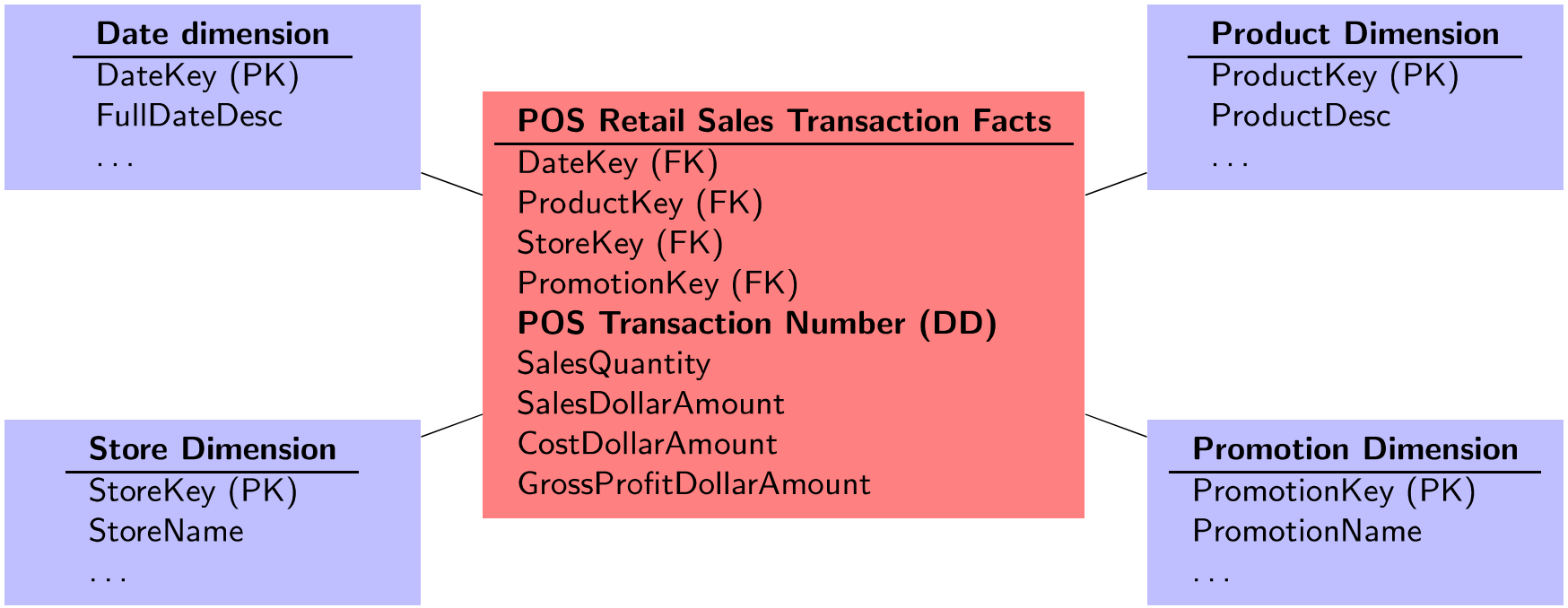
Product Dimension
Product Key (PK)
Product Description
SKU Number
Brand Description
Category Description
Department Description
Package Type Description
Package Size
Fat Content
Diet Type
Weight
Weight Units of Measure
...

Instance of Product dimension table

PK	Product Description	Brand Desc	Cat Desc	Dept Desc	...
1	Baked Well Light	Baked Well	Bread	Bakery	
2	Fluffy Sliced Whole Wheat	Fluffy	Bread	Bakery	
3	Fluffy Light Sliced Whole Wheat	Fluffy	Bread	Bakery	
4	Fat Free Mini Cinnamon Rolls	Light	Sweeten Bread	Bakery	
5	Diet Lovers Vanilla 2 Gallon	Coldpack	Frozen Desserts	Frozen Foods	
6	Light and Creamy Butter Pecan 1 Pint	Freshlike	Frozen Desserts	Frozen Foods	

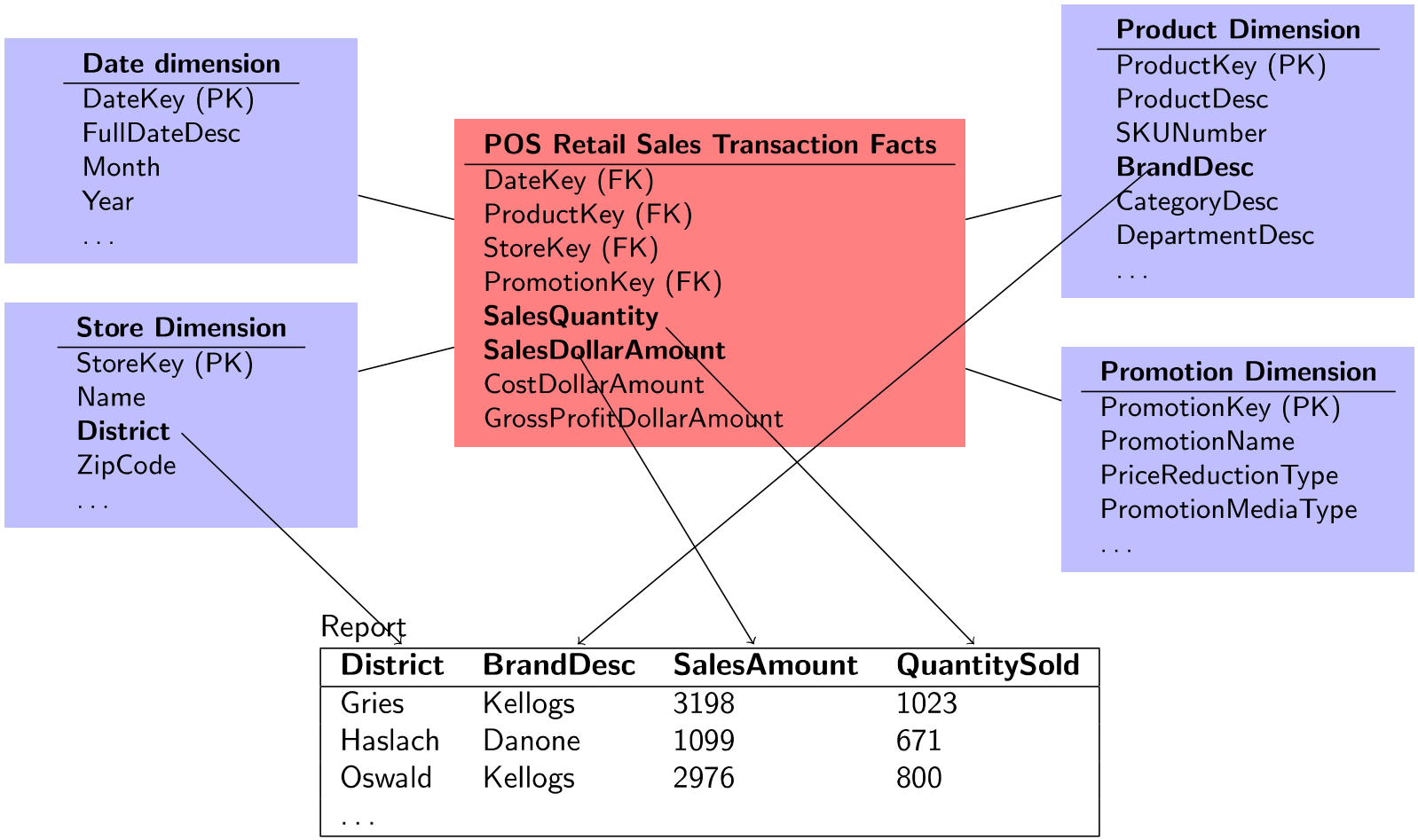
Degenerate Dimensions

- Degenerate dimensions are "empty"
  - i.e., dimensions with a "hierarchy" of only one attribute
  - Values are directly stored in fact table, no dimension table is needed
- Examples are operational control numbers, e.g., order #, invoice #, POS transaction #, etc.
- 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



Working with a Dimensional Model/1

- Each dimension attribute is a rich source for constructing row headers
- A common activity is to drag and drop dimensional attributes and measures into a simple report (+ specification of aggregate functions for measures)



District	BrandDesc	SalesAmount	QuantitySold
Gries	Kellogs	3198	1023
Haslach	Danone	1099	671
Oswald	Kellogs	2976	800
...			

Working with a Dimensional Model/2

- Drilling down is adding row headers from a dimension
- Rolling up is removing row headers

Dept desc	Sales Dollar Amount	Sales Quantity
Bakery	$12.331	5088
Frozen Foods	$31.776	15.565

Roll up on Product dimension

Drill down on Product dimension

Product Dimension
ProductKey (PK)
ProductDesc
SKUNumber
BrandDesc
CategoryDesc
DepartmentDesc
...

Dept Desc	Brand Desc	Sales Dollar Amount	Sales Quantity
Bakery	Baked Well	$3.009	1.138
Bakery	Fluffy	$3.024	1.476
Bakery	Light	$6.298	2.474
Frozen Foods	Coldpack	$5.321	2640
Frozen Foods	Freshlike	$10.476	5.234

Outline

1 The Grocercy Store Example

2 More about Multidimensional Modeling

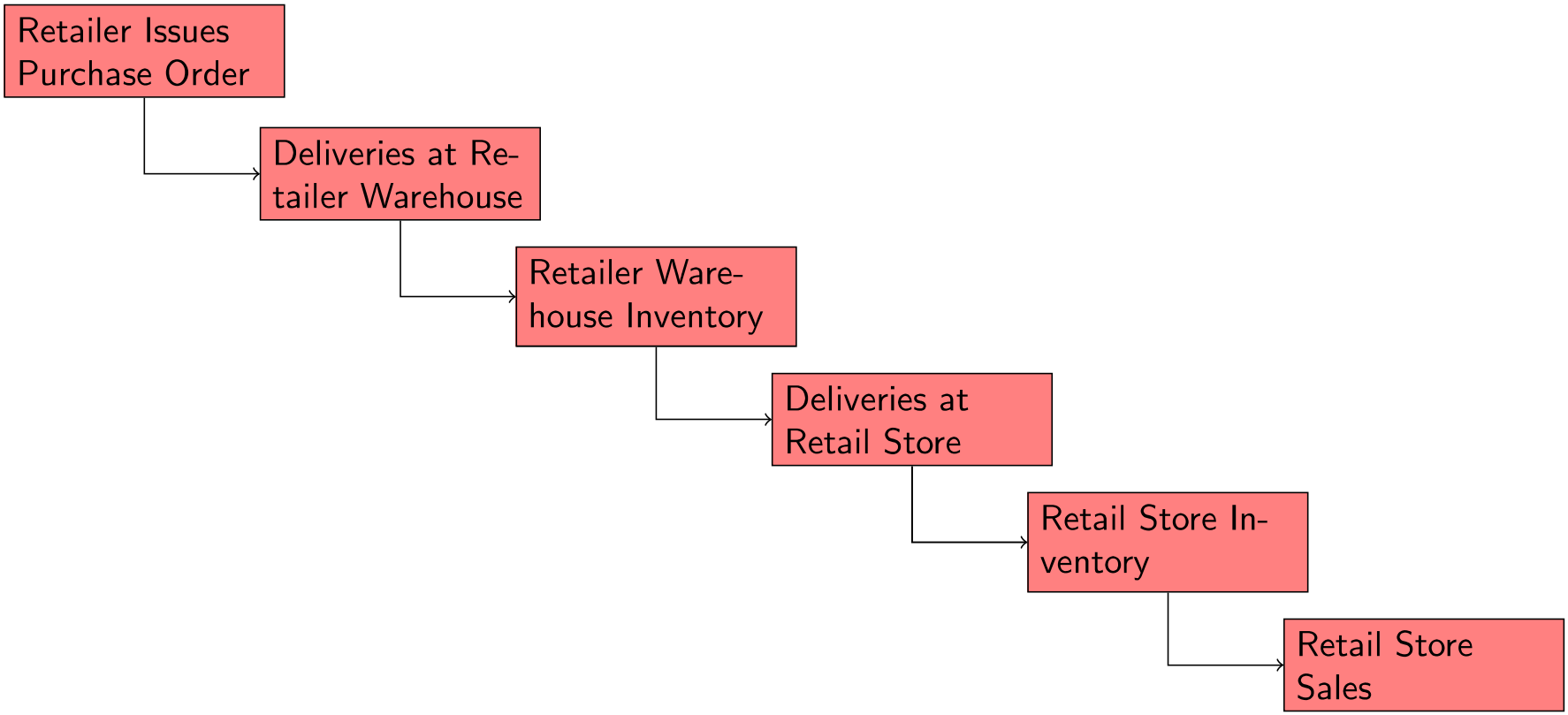
3 Inventory Management Example

Inventory Management

- Consider a large grocery chain with a central warehouse and several retail stores
- Advanced retail business requires inventory information
  - Making sure the right product is in the right store at the right time minimizes out-of-stocks and reduces overall inventory carrying costs
  - The retailer needs the ability to analyze daily quantity-onhand inventory levels by product and store
- Design dimensional models that support the analysis of inventories for retail businesses (grocery stores)

The Value Chain

- The value chain identifies the natural, logical flow of an organization's primary activities
- Provides useful information for the identification of business processes
- Operational systems provide snapshots at each step with interesting data and performance metrics



Inventory Models

- 3 different inventory models
  - Model 1: Inventory periodic snapshot model
  - Model 2: Inventory transactions model
  - Model 3: Inventory accumulating snapshot model

Inventory Periodic Snapshot Model/1

- **Model 1: Inventory Periodic Snapshot**: Every day (or at some other regular time interval) the inventory levels of each product is measured and stored as a new row in the fact table
- **Example:** Inventory of retail store
  - Business process: Analysis of retail store inventory
  - Granularity: Daily inventory by product at each individual store
  - Dimensions: Date, product, and store
  - Facts/measures: Quantity on hand

Inventory Periodic Snapshot Model/2

- Inventory generates dense snapshot tables (i.e., entry for each product)
  - In contrast, POS Retail Sales table was sparse
- Hence, inventory fact table is growing fast!
  - 60.000 products $\times$ 100 stores $=$ 6 Mio. rows each time
  - With a row width of 14 bytes, this is 84 MB each time
  - 1 year of daily snapshots would be 30 GB
- Reduce snapshot frequencies over time
  - Last 60 days of inventory at daily level
  - Weekly snapshots for older data
  - Instead of 1.095 snapshots in 3 years, only 208 snapshots would be required

Inventory Periodic Snapshot Model/3

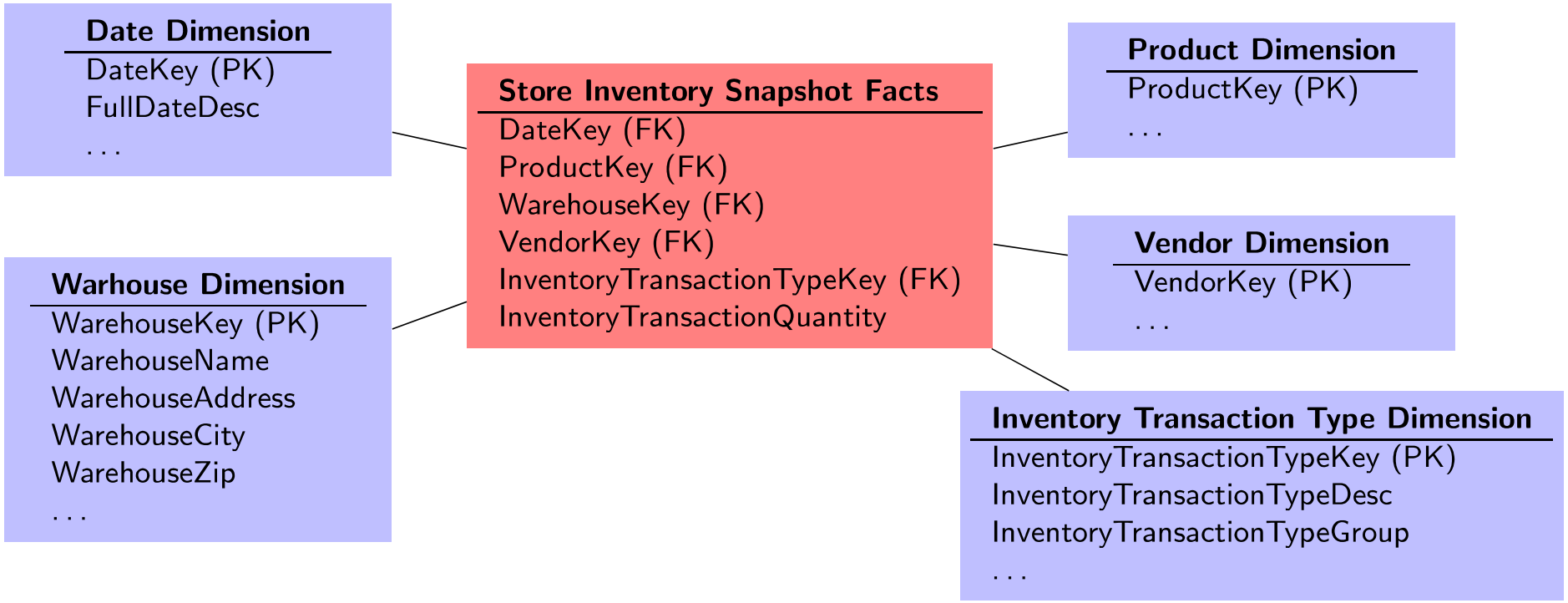
- The quantity on hand is a semi-additive measure
  - Can be summarized across products and stores, but not across time
  - Different in POS Retail Sales table: Sold entities are counted only once
- All measures that record a static level (inventory, financial account balance, measures of intensity, e.g., temperature) are inherently non-additive across time and possibly other dimensions
  - Can be aggregated along time dimension by averaging
- A note about SQL AVG function:
  - Cannot be used to compute the average over time, since it averages over the number of rows
  - Avg inventory over a cluster of 3 products in 4 stores across 7 days would divide the summed value by 84

Inventory Transactions Model/1

- **Model 2: Inventory Transactions:** Every transaction that affects the inventory is recorded
- **Example:** Inventory transactions in the store chain
  - Receive product
  - Place product in to inspection hold
  - Release product from inspection hold
  - Return product to vendor due to inspection failure
  - Place product in bin
  - Authorize product for sale
  - Pick product for shipment
  - Ship product to customer
  - Receive product form customer
  - Return product to inventory from customer return
  - Remove product from inventory

Inventory Transactions Model/2

- Star schema of the inventory transaction model

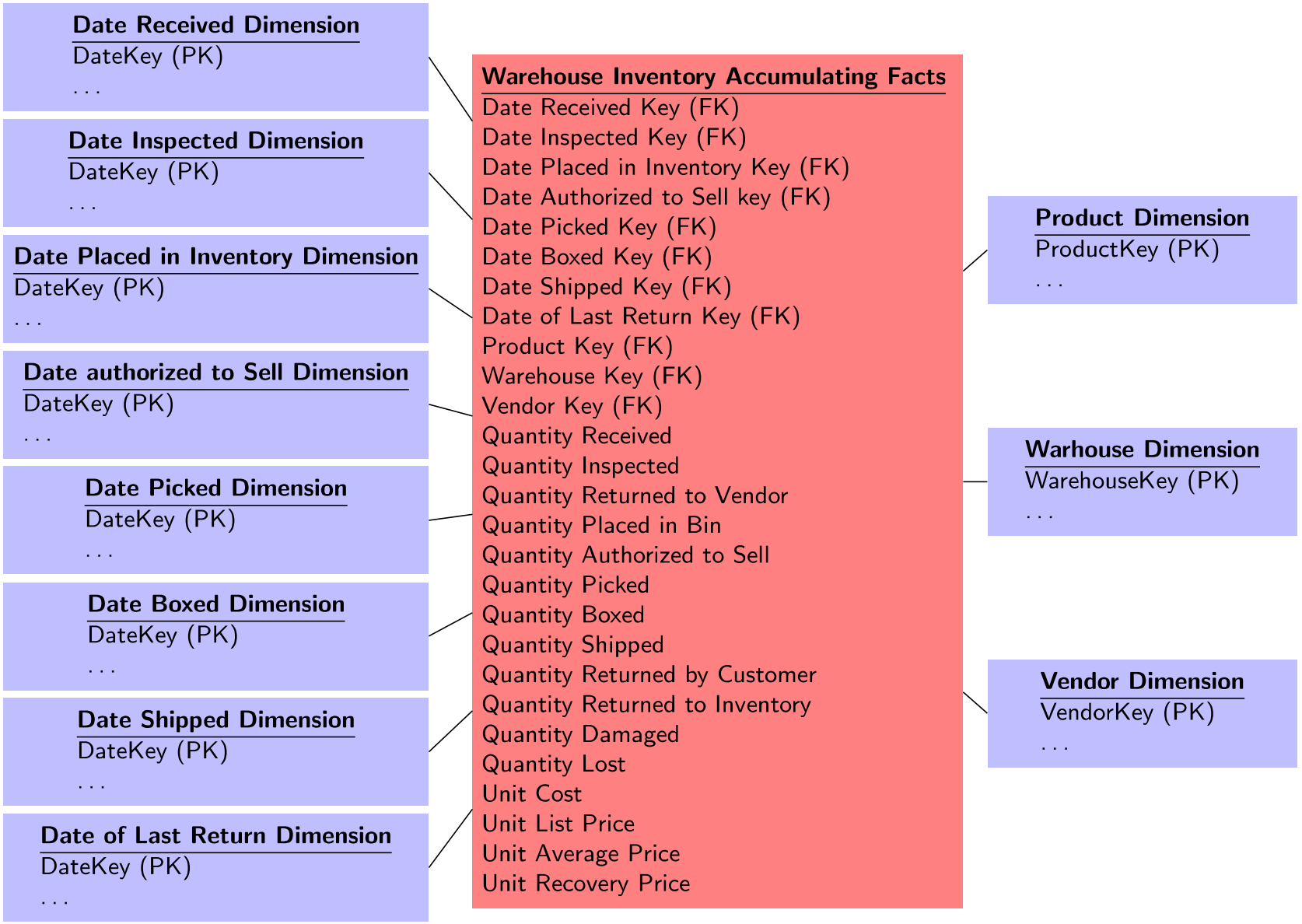


- Contains most detailed information, e.g.,
  - How many shipments from a given vendor?
  - On which products more than one round of inspection?
- Reconstruction of exact inventory numbers is possible, but not practical!
  - Used in combination with other fact table

Inventory Accumulative Snapshot Model/1

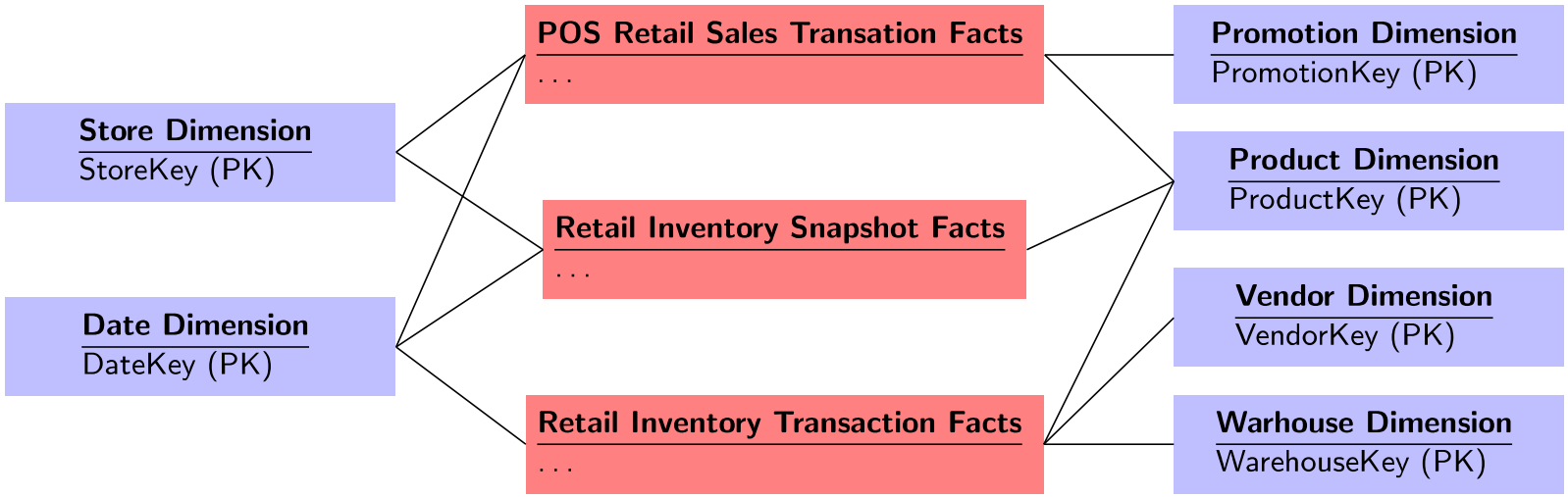
- **Model 3: Inventory Accumulating Snapshot:** One row in the fact table for each shipment of a particular product to the warehouse
- Assumption that the inventory goes through a series of events, e.g., receiving, inspection, bin placement, authorization to sell, picking, boxing, and shipping.
- A row tracks the disposition of a shipment through these events in the warehouse.
- Row is updated as the shipment moves through the warehouse until it leaves the warehouse.
- Characterized by many date dimensions and many updates.

Inventory Accumulative Snapshot Model/2



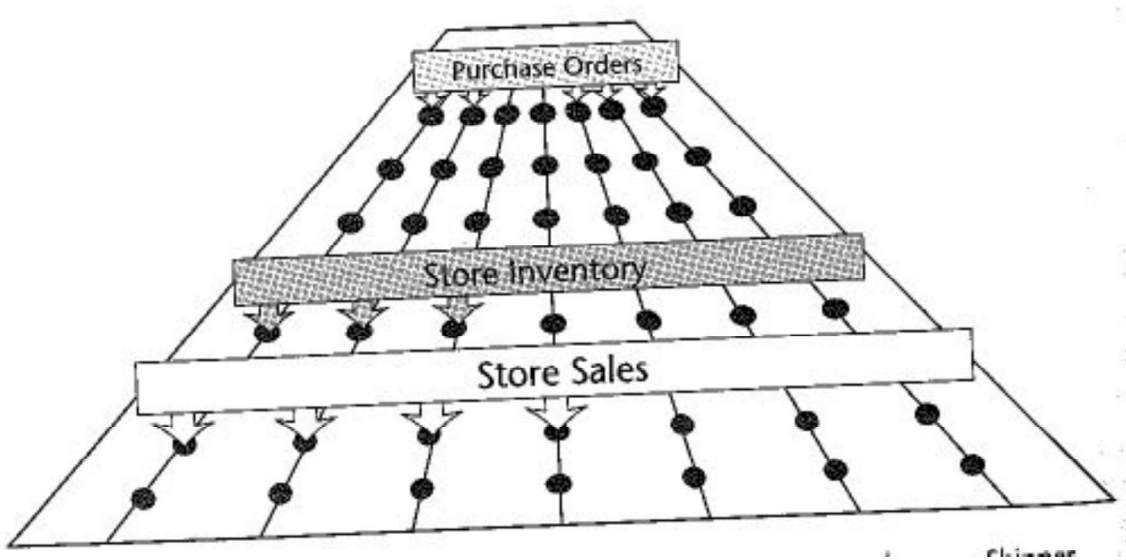
Value Chain Integration and Shared Dimensions

- Integration accross the value chain is important for the analysis
  - Allows analysis across the business to better evaluate the overall performance (not just at the individual department level)
  - End-to-end perspective high-level management to customer
- This requires the integration and consistent handling/use of data
- Solution: Individual fact tables for processes + shared dimensions
- **Shared dimensions** are used by different data marts



Data Warehouse Bus Architecture

- **Data Warehouse Bus Architecture** is a standard bus interface that supports the incremental development of a DW
- Based on conformed (similar) dimensions that are shared by the DMs
- Useful tool for the design process as it breaks down the process into small chunks (DMs)
- DMs can be realized at different times and by different groups



Data Warehouse Bus Matrix/1

- **Data Warehouse Bus Matrix** is a way to document the bus architecture
  - Rows represent business processes (translate into DMs)
  - Columns represent a suite of standardized, common and shared dimensions

	Shared Dimensions							
Business Processes	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

Data Warehouse Bus Matrix/2

- Creating the DW bus matrix is one of the most important up-front deliverables of a DW implementation
- Create a comprehensive list of dimensions before filling in the matrix
- The rows provide a concise overview about the dimensionality of the individual DMs
- The columns show the interaction between the DMs and the common/shared dimensions

Summary/1

- Simplified DW design process by Kimball and Ross consists of 4 steps:
  - Choose business processes, granularity, dimensions, and measures
- Surrogate key should be used instead of operational codes
- Degenerate dimensions are stored in the fact table
  - or stored all together in a junk dimension
- DW sizing
  - dimensions are a small portion of the DW, hence can/should contain as much information as possible
  - fact table determines the size of the DW
- Dimensional model is easy to use, e.g., drag and drop for report generation

Summary/2

- Different Inventory models: periodic snapshot model, transactions model, accumulating
- Shared and conformed dimensions are crucial to integrate several DMs across a value chain
- DW bus architecture is a standard interface to support incremental DW design
- DW bus matrix is a way to document the DW bus architecture
- Role-playing allows to physically store a dimension only once, but use it several times in different roles and with different names
- Fact normalization collapses all measures into a single measure together with a special fact dimension to determine the type of the measure
  - Only useful when the fact table is sparse
- Multi-valued dimensions if a dimension occurs more than once in a single fact, e.g., a patient has typically several diagnoses
  - Corresponds to multiple arcs in DFM (many-to-many relationships)
- Bridge tables can be used to represent such dimensions