Data Warehousing and Data Mining

Lecturer: Dr Patrick Pang

patrickpang@ipm.edu.mo

Ch1

OLAP (Online Analytic Processing)

Ch2 Require analysis, Conceptual design

Requirement Analysis

it is significant because it influences many things

- Collect end user needs for data mart applications and usage
- Model the information needs of end users using the multidimensional data model

Concepts

- **Facts** are concepts on which end users base their decision-making process. Facts refer to a category of events taking place in the business.
  - E.g. sales, shipments, hospital admissions, surgeries
- Instances of a fact correspond to **events** that occurred.
  - E.g. every single sale or shipment carried out is an event
- Each fact is described by the values of a set of relevant **measures** that provide a quantitative description of events.
  - E.g. sales receipts, amounts shipped, hospital admission costs, and surgery time
- The large number of events may be selected and sorted out by different **dimensions**.
  - E.g. Sales in a store chain can be represented in a 3D space whose dimensions are products, stores (geography), and dates (time).

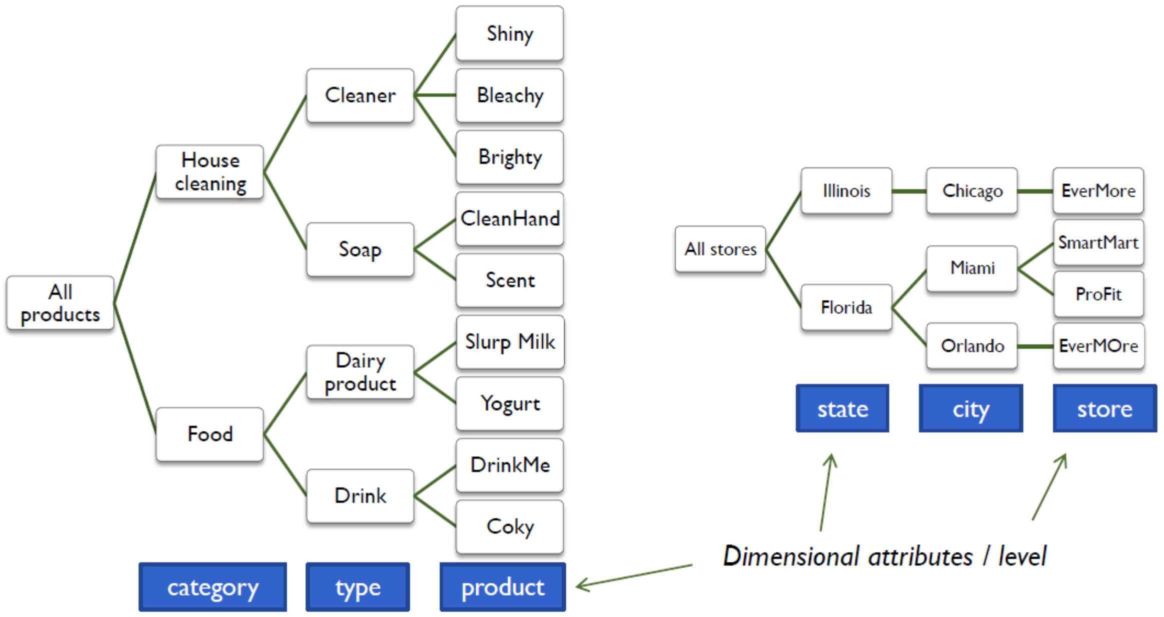
Dimensions

Each dimension is further described by many dimensional attributes

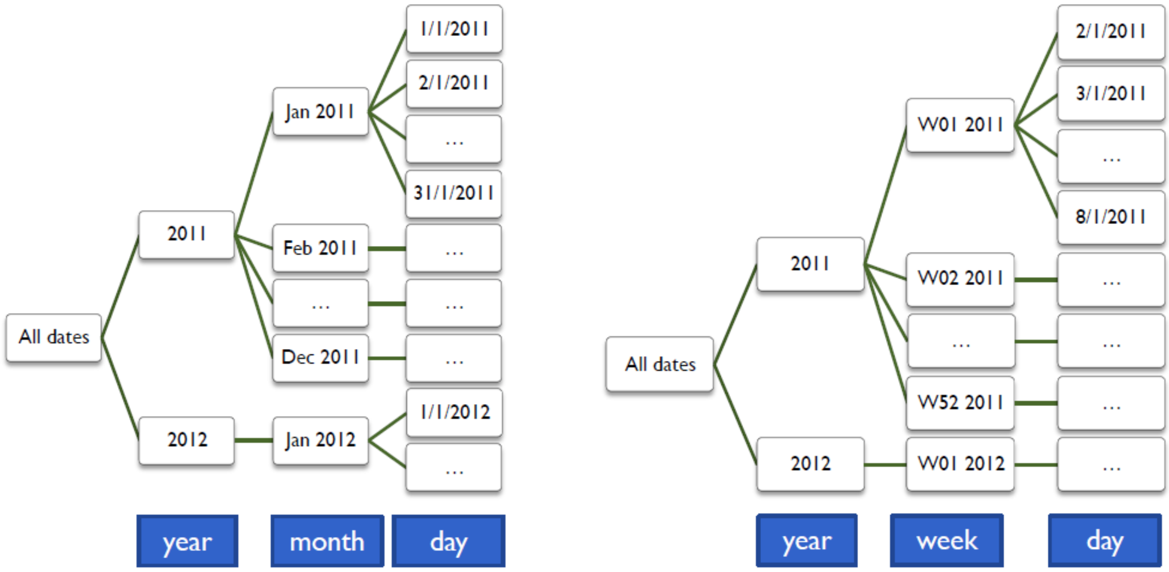
Also commonly known as levels

Examples:

- Levels for dimension 'Date': date, month, year
- Levels for dimension 'Product': product, type, category
- Levels for dimension 'Store': store, city, state



Dimensional attributes / level



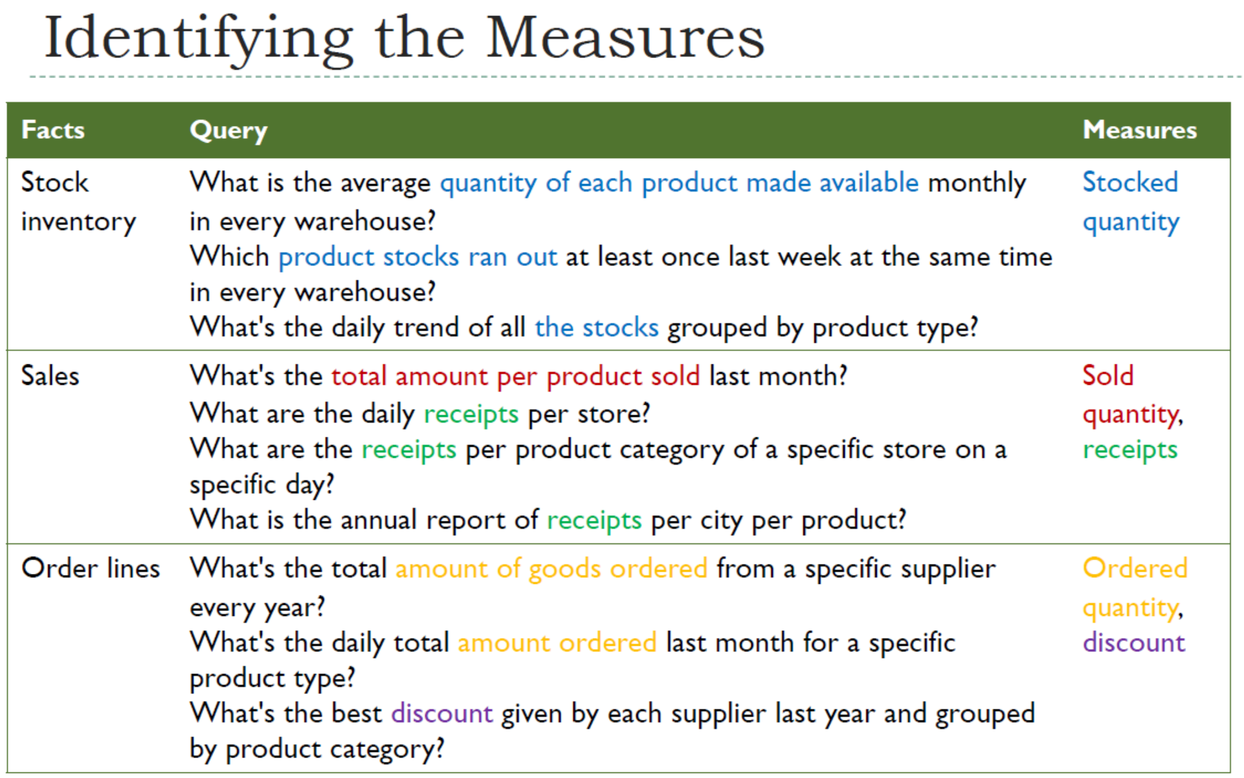
A dimension may have more than 1 hierarchy that organizes the instances in the dimension in different ways

(P.S. Finest granularity: one day)

Reduce the amount of data

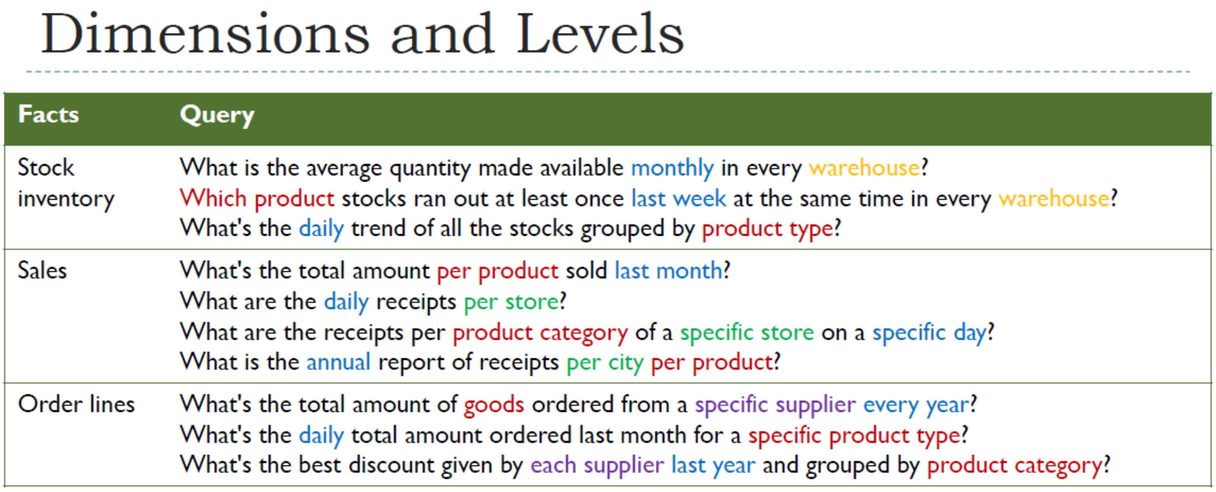
- **Restriction**: e.g. show only sales data for last month in stores in a city
- **Aggregation**: e.g. show total sales receipts by product types and by city (of the store)

Both restriction and aggregation are achieved with levels



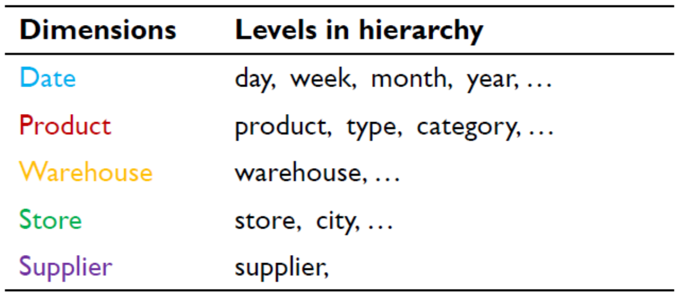
Identifying the Measures

Facts	Query	Measures
Stock inventory	What is the average quantity of each product made available monthly in every warehouse? Which product stocks ran out at least once last week at the same time in every warehouse? What's the daily trend of all the stocks grouped by product type?	Stocked quantity
Sales	What's the total amount per product sold last month? What are the daily receipts per store? What are the receipts per product category of a specific store on a specific day? What is the annual report of receipts per city per product?	Sold quantity, receipts
Order lines	What's the total amount of goods ordered from a specific supplier every year? What's the daily total amount ordered last month for a specific product type? What's the best discount given by each supplier last year and grouped by product category?	Ordered quantity, discount



Dimensions and Levels

Facts	Query
Stock inventory	What is the average quantity made available monthly in every warehouse? Which product stocks ran out at least once last week at the same time in every warehouse? What's the daily trend of all the stocks grouped by product type?
Sales	What's the total amount per product sold last month? What are the daily receipts per store? What are the receipts per product category of a specific store on a specific day? What is the annual report of receipts per city per product?
Order lines	What's the total amount of goods ordered from a specific supplier every year? What's the daily total amount ordered last month for a specific product type? What's the best discount given by each supplier last year and grouped by product category?



Dimensions	Levels in hierarchy
Date	day, week, month, year, …
Product	product, type, category, …
Warehouse	warehouse, …
Store	store, city, …
Supplier	supplier,

Conceptual Modeling

A conceptual model captures concepts and relationship between them

Although ER diagram is expressive enough to represent most concepts in multidimensional data, it cannot accurately highlight the distinctive features of the data.

Some designers skip conceptual modeling and base their design on a logical model like the star schema. However,

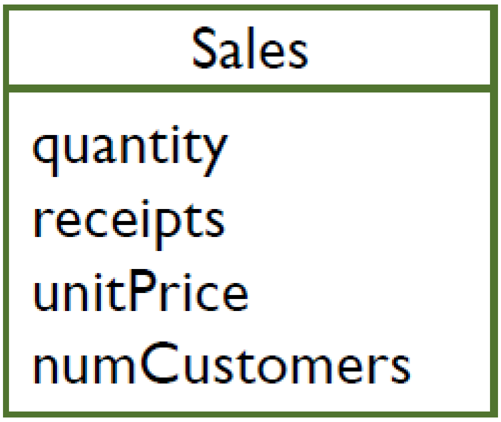
- It is less responsive to requirements, harder to maintain and reuse
- Denormalizaton of dimension hides some functional dependencies

We will use the Dimensional Fact Model (DFM) to do conceptual design

Concepts

Fact and measure

- A fact is a concept relevant to decision-making processes. It typically models a set of events taking place within a company
  - A fact have dynamic properties or evolve in some way over time
- A measure is a numerical property of a fact and describes a quantitative fact aspect that is relevant to analysis
  - Measures are preferably numeric because it is easy to make calculation
  - Need to consider how to obtain measure values from operational data
  - An empty fact schema has no measures and records only occurrence of an event.

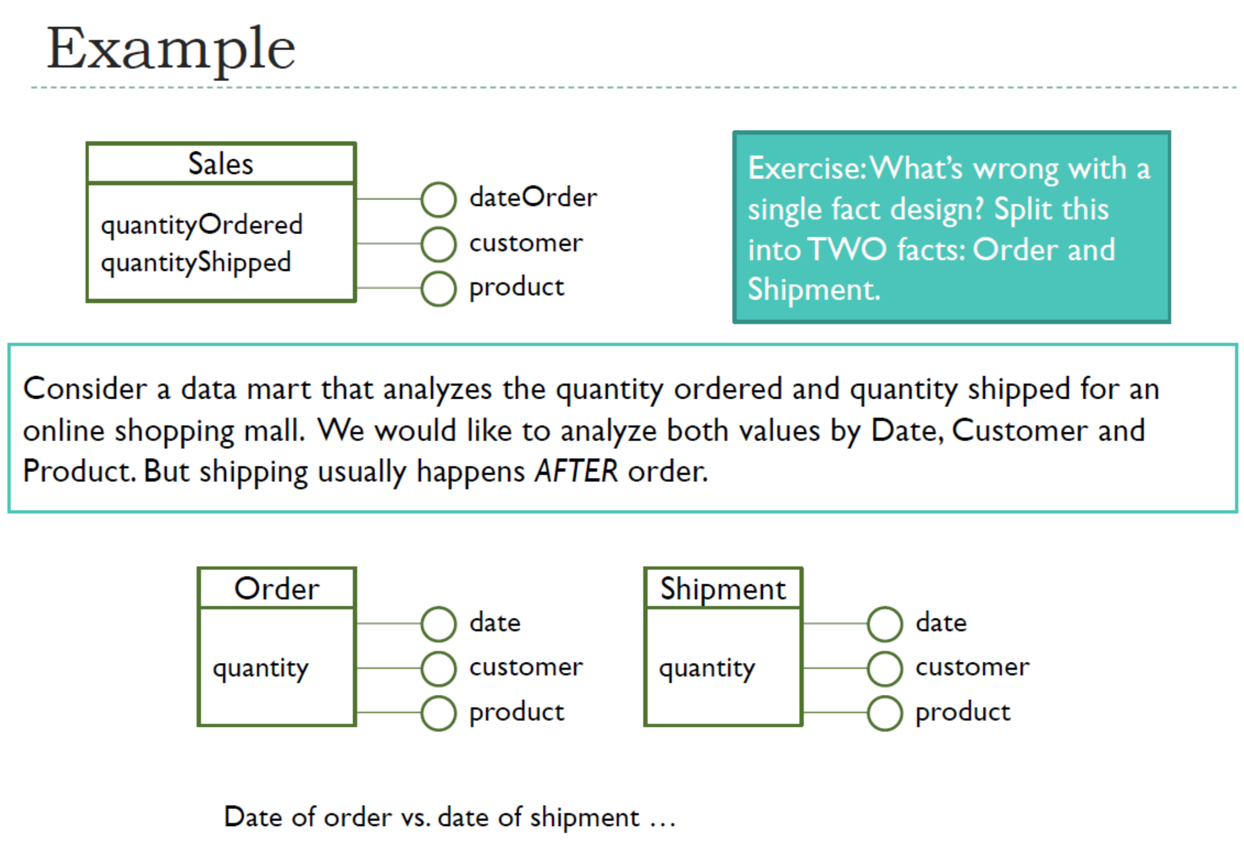


One or More Facts?

To determine whether two measures belong to the same fact, consider two questions:

- Do these measures occur simultaneously?
- Are these measures available at the same level of detail?

If either is ‘false’, the two measures belong to separate facts.


Example

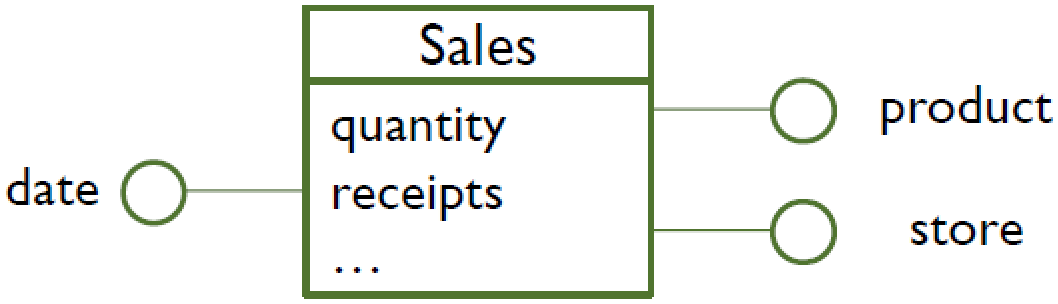
Exercise: What's wrong with a single fact design? Split this into TWO facts: Order and Shipment.

Consider a data mart that analyzes the quantity ordered and quantity shipped for an online shopping mall. We would like to analyze both values by Date, Customer and Product. But shipping usually happens *AFTER* order.

Date of order vs. date of shipment …

Dimension

- A dimension describes an analysis coordinate of the fact.
- The dimensions of a fact define its minimum representation **granularity**.
  - E.g. The fact Sales, which has the dimensions product, store and date, can represent product sales in one store in one day. It cannot distinguish sales made by different customers or at different times of day.
- The **grain statement** describes the meaning of an event in the fact
  - E.g. On a certain date, some customers bought a certain product at a certain store. The number of items sold is the measure 'quantity' and the total sales account in dollar is the measure 'receipts'.
- Because facts are generally dynamic, a fact schema will almost certainly have at least one **temporal dimension**.



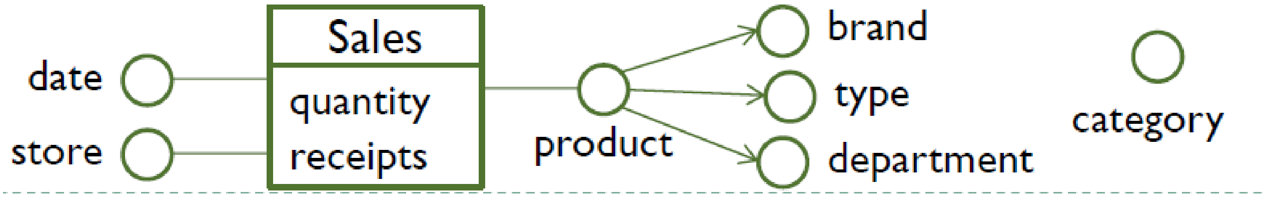
Conformed Dimension

To perform analysis across multiple facts, the facts must use conformed dimension.

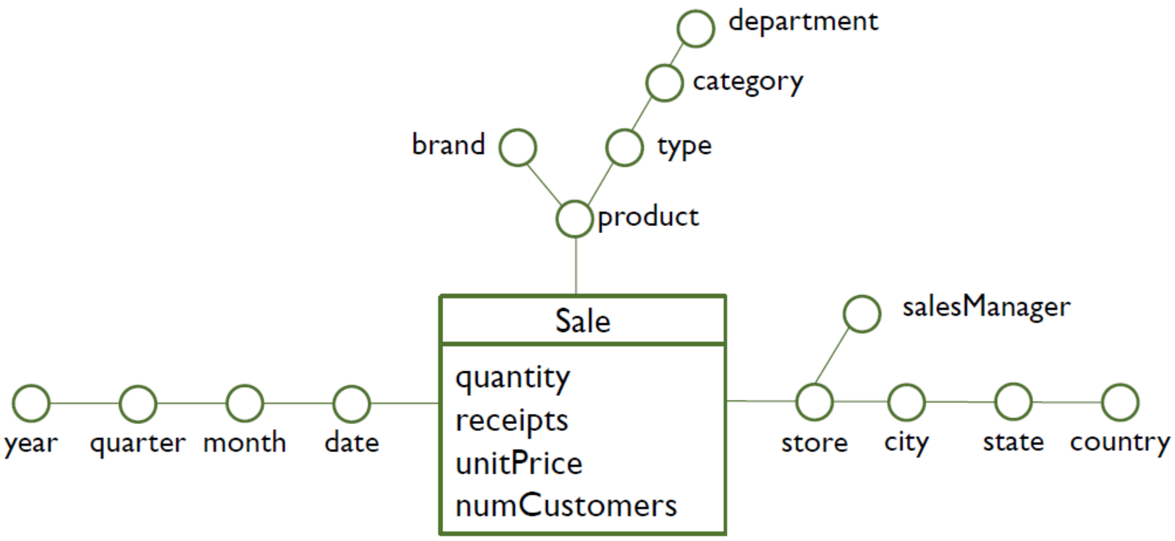
The shared dimensions must have same structure and same value.

Dimensional attributes

- Dimensional attributes describe instances of a dimension
  - E.g. the product 'coca-cola zero' belongs to the type 'soft drink', which is under the category 'drink'. The product has a brand 'coca-cola', and is sold in the department 'soft drink' of the supermarket.
- We use a unique name for the instances to represent the dimension in the schema. This name is also regarded as a dimensional attribute
  - E.g. the attribute product (sample value 'coca-cola zero') in the product dimension, or the attribute date (sample value '2015-01-22') in the date dimension
- Some dimensional attributes have many-to-one association. This is modeled as functional dependency in the relational model.
  - E.g. a specific product belongs to one brand, while one brand is associated with multiple products. The FD is product à brand



- Dimensional attributes of a dimension can be modeled as a directed tree whose nodes are dimensional attributes and whose arcs model many-to-one associations (functional dependency) between dimensional attribute pairs.
  - If there are no ambiguity, we may omit the arrow in the lines between dimensional attributes.

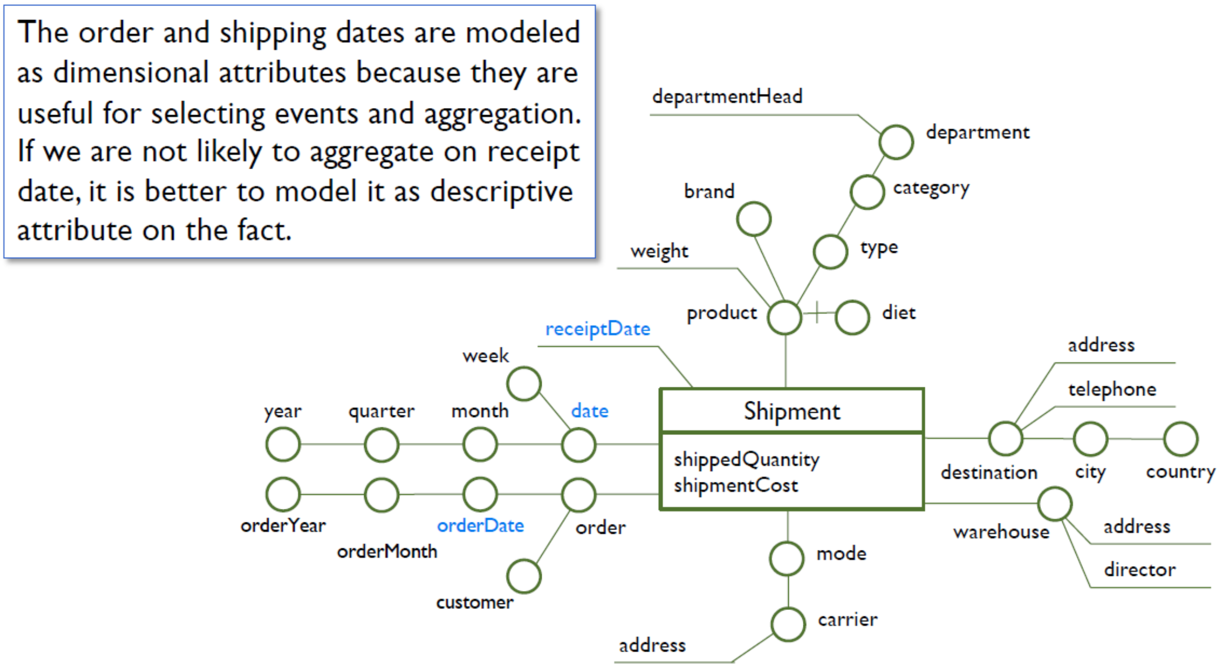


- Dimensional attributes usually have descriptive textual values

- E.g. 'Jan 2013', '$10k-$20k', 'Beijing, China'
- Simple to use as labels in reports and in SQL queries
- Flags (boolean attributes) should be coded as readable strings, e.g. 'credit order' and 'not credit order'

Descriptive Attributes

- Descriptive attributes are additional information that are not likely to be used as aggregation criteria
  - may appear on a dimensional attribute in a hierarchy or a fact
  - may be string values (e.g. store address)
  - may take continuous values (e.g. weight = 1.23kg) and won't be used for aggregation
- A descriptive attribute on a dimensional attribute is functionally determined by the dimensional attribute
  - E.g. weight of a product is fixed for a specific product
- A descriptive attribute on a fact describes a primary event

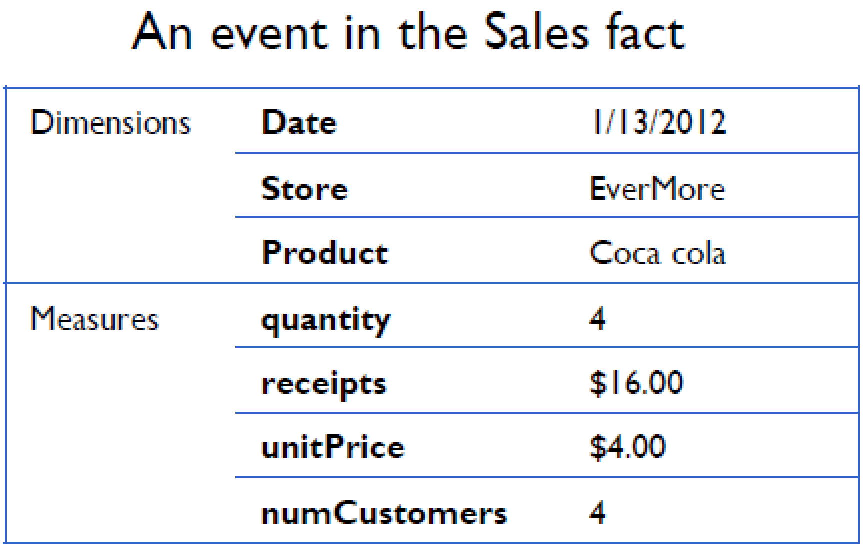


The order and shipping dates are modeled as dimensional attributes because they are useful for selecting events and aggregation. If we are not likely to aggregate on receipt date, it is better to model it as descriptive attribute on the fact.

Primary event

A primary event is a particular occurrence of a fact, identified by one tuple made up of a value for each dimension. A value for each measure is associated with each primary event.

- Depending on granularity of the fact schema, a primary event may not have one-to-one relationship with discrete events in the business.



An event in the Sales fact

Dimensions	**Date**	1/13/2012
	Store	EverMore
	Product	Coca cola
Measures	**quantity**	4
	receipts	$16.00
	unitPrice	$4.00
	numCustomers	4

Workload refinement

Verify the conceptual schema against the preliminary workload.

Some common problems in the verification:

- Cannot aggregate data at the required level because of lacking dimensional attributes (e.g. in the Date dimension)
- A false functional dependency between the attributes prevents a valid group-by set that can satisfy an analysis query (e.g. a FD between course and teacher prevents comparison of student performance taking the same course taught by different teachers)
- User cannot select events the way they want to, because of lacking dimensional attributes (e.g. total sales during the CNY holiday in last 5 years, sales by credits vs. cash)
- Missing measure. Can we derive it from existing measures or data in the operational data? Should we store it in primary events?

Fact Schema

Transactional fact schemas

- Each primary event summarizes 1 or more activities in a business process
- No primary event created if there are no corresponding business transaction
- Use SUM to summarize the measures

Snapshot fact schemas

- Each primary event records the status measurement done periodically
- Primary events have no direct correspondence with business transaction
- Cannot use SUM in some situations

Transactional Fact Schema

A transactional fact schema tracks the individual activities that define a business process and supports several measures that describe these activities.

Properties of transactional fact schema
Grain: Lossy and lossless, Sparse

Additive measures

- Easy to summarize in rollup
- Pre-calculated aggregates (materialized view)

Handling non-additive measures

- Ratio and Average: use stored additive components to calculate in query time
- Use MIN,MAX, AVG to summarize non-additive measures

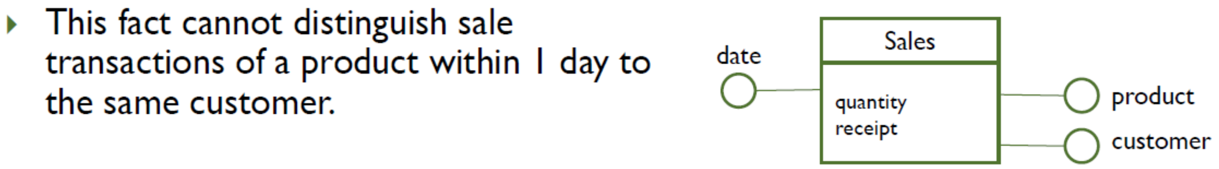
Lossy-grained transactional fact

In a lossy-grained fact, a primary event summarizes 1 or more business transactions

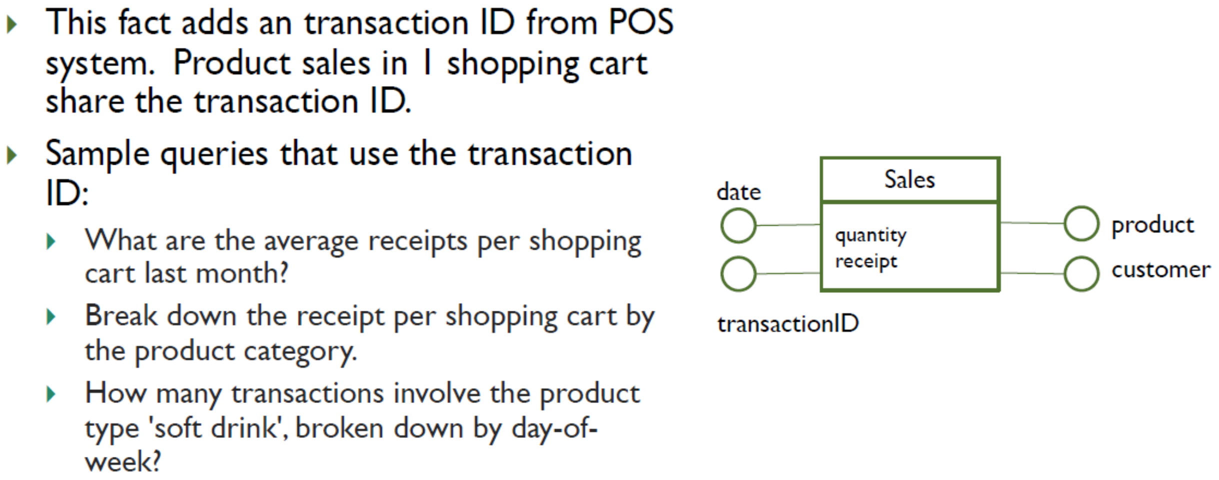
Lossless-grained transactional fact

In a lossless-grained fact, a primary event records detail of each business transaction
E.g. the Sales fact on the left records the sale of a certain product at a specific time on a certain date in a certain store sold to a certain customer

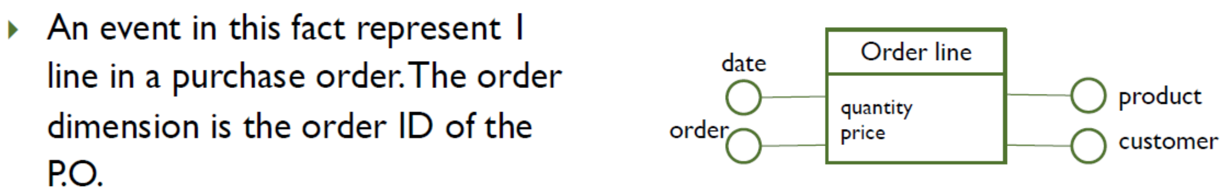
Fine granularity: usually requires a timestamp and/or transaction ID



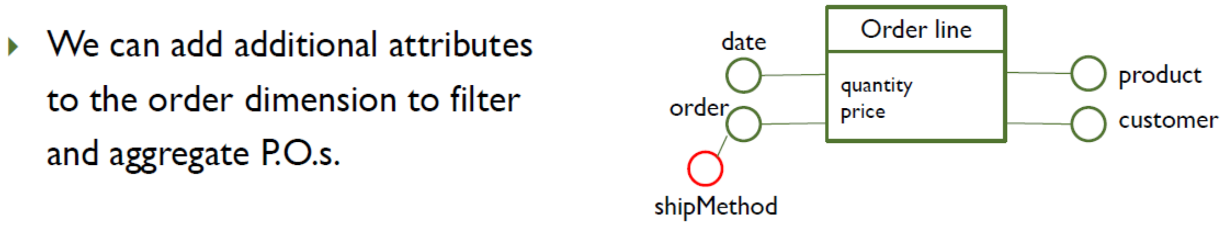
- This fact cannot distinguish sale transactions of a product within 1 day to the same customer.



- This fact adds an transaction ID from POS system. Product sales in 1 shopping cart share the transaction ID.
- Sample queries that use the transaction ID:
  - What are the average receipts per shopping cart last month?
  - Break down the receipt per shopping cart by the product category.
  - How many transactions involve the product type 'soft drink', broken down by day-of-week?



- An event in this fact represent 1 line in a purchase order. The order dimension is the order ID of the P.O.



- We can add additional attributes to the order dimension to filter and aggregate P.O.s.

Additive Measures

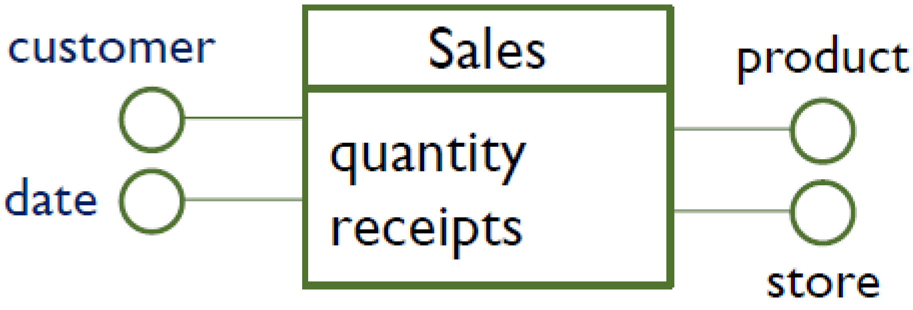
Measures in transactional facts are usually additive

A measure is called additive when you can use the SUM operator to aggregate along any dimension.
i.e. the sum of measures of a group of events are meaningful

Generally, we store additive measures in primary events.
Measures that sum up count or dollar amount in a business

transaction are usually additive

- total quantity of apples sold in a store yesterday,
- total sales receipts from each customer in a store yesterday,
- total sales receipts from selling fruits in each store last month



Non-additive measures

Business users also request non-additive measures in analysis queries.

Common examples: ratio and average

- Ex.1 margin rate = ('total order dollars' – 'total cost dollars') / 'total order dollars'
  Storing the margin rate in primary events is not useful adding the margin rate of several transactions does not give the margin rate of the transactions.
- Ex.2 average order dollars = 'total order dollars' / 'total quantity ordered'
  Storing the average order dollars of each transaction is not useful either.

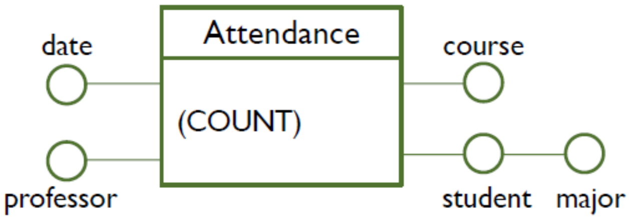
Basic strategy: store additive components in primary events, and **calculate non-additive measures at query time**. These are called calculated measures.

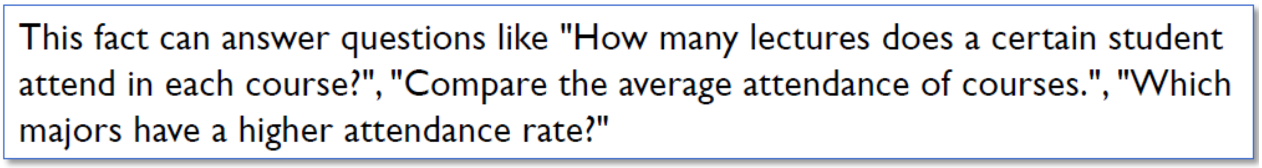
- Store the additive components as measures in fact schema
- Do not store the ratio / percentage directly!

Empty Fact Schema

An empty fact schema does not have any measures

Each primary event records the occurrence of an activity, e.g. customer service request, click-thru count of online advertisement, web page view




This fact can answer questions like "How many lectures does a certain student attend in each course?", "Compare the average attendance of courses.", "Which majors have a higher attendance rate?"

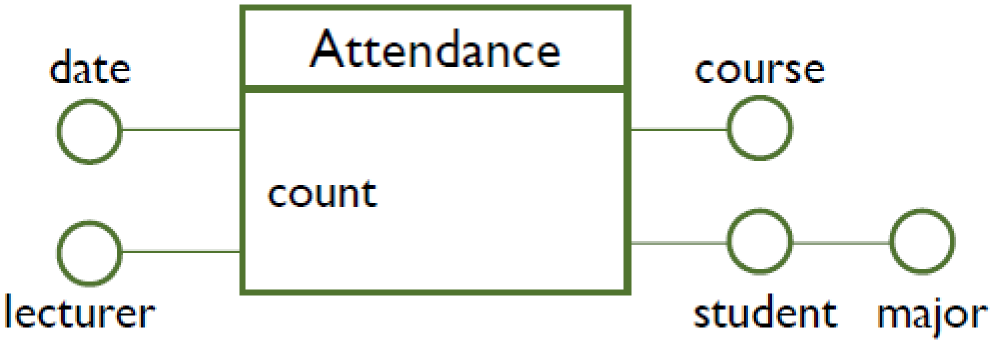
Alternative way

add a count measure and initialize it to 1 for all primary events

Count is additive

When aggregate the data, just calculate the total count.
E.g. "how many lectures does a certain student attend in each courses" becomes "the total count of attendance of a certain student in each course".

Better support in OLAP engines...



Snapshot Fact Schema

Ch3 Logical design

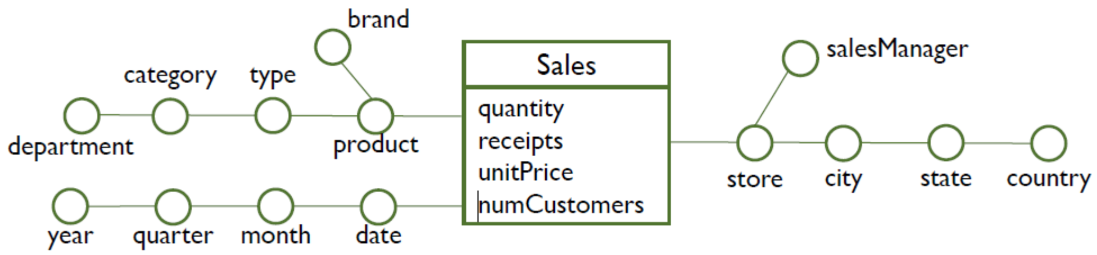
In logical design, a data mart designer has to select a logical model to implement the conceptual schema.

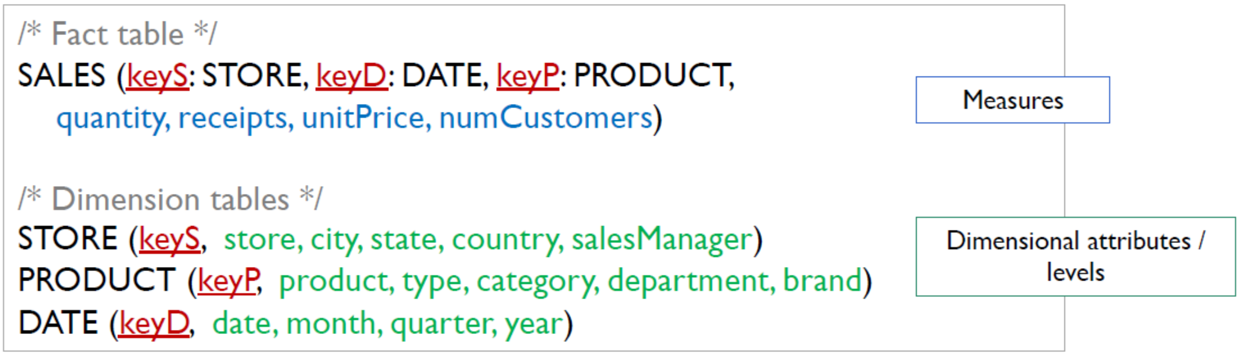
Three common approaches

- Relational OLAP (ROLAP) uses relational model in common DBMS to represent the multidimensional data
- Multidimensional OLAP (MOLAP) implements the multidimensional data structure natively in a database
- Hybrid OLAP (HOLAP) hybrid model stores a part of the data in multidimensional data structure

Star schema

ROLAP uses a star schema to represent multidimensional data.





/* Fact table */
SALES (keyS: STORE, keyD: DATE, keyP: PRODUCT,
quantity, receipts, unitPrice, numCustomers)

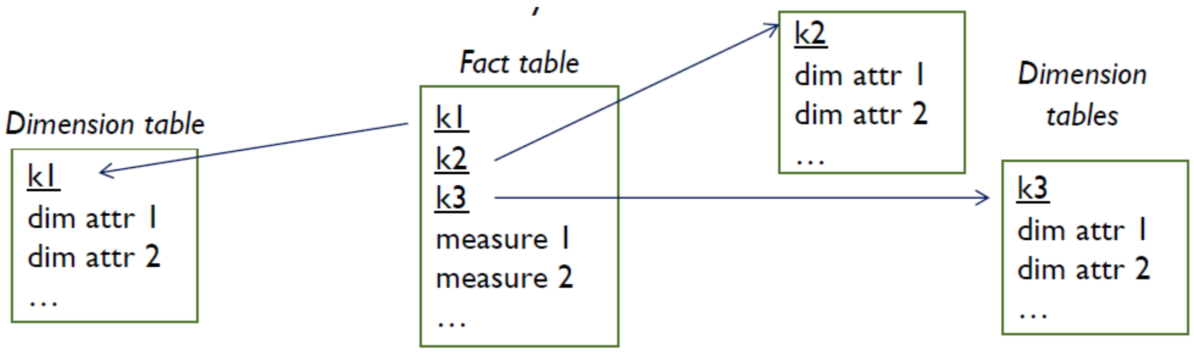
/* Dimension tables */
STORE (keyS, store, city, state, country, salesManager)
PRODUCT (keyP, product, type, category, department, brand)
DATE (keyD, date, month, quarter, year)

Measures

Dimensional attributes / levels

A star schema consists of the following:

- One dimension table for each dimension
  - Each dimension table features a primary key (ki) and a set of attributes describing its dimension instance at different aggregation levels
- One fact table referencing all the dimension tables
  - Its primary key is the composition of the set of foreign keys (k1 through kn) referencing the dimension tables.
  - t includes an attribute for every measure.



Why Star

Dimension tables are denormalized (i.e. not normalized)

Some benefits:

- Star schema is easier to understand
- Redundancy in the data set simplifies the ETL process. (less primary key / foreign key relationship to maintain)
- Better query performance because of smaller number of table joins

Create a star schema from the fact schema,

Create a dimension table for each dimension

- Add a surrogate key to each dimension table
- Add an attribute for each dimensional attributes (i.e. levels) in the dimension

Create a fact table for the fact

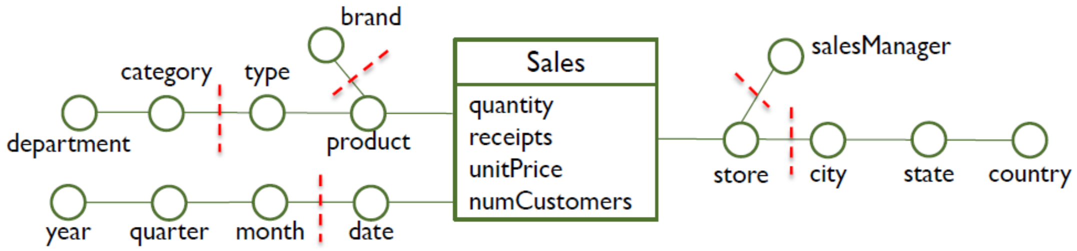
- The composite key of the fact table consists of foreign keys to each dimension tables
- Add an attribute for each stored measure

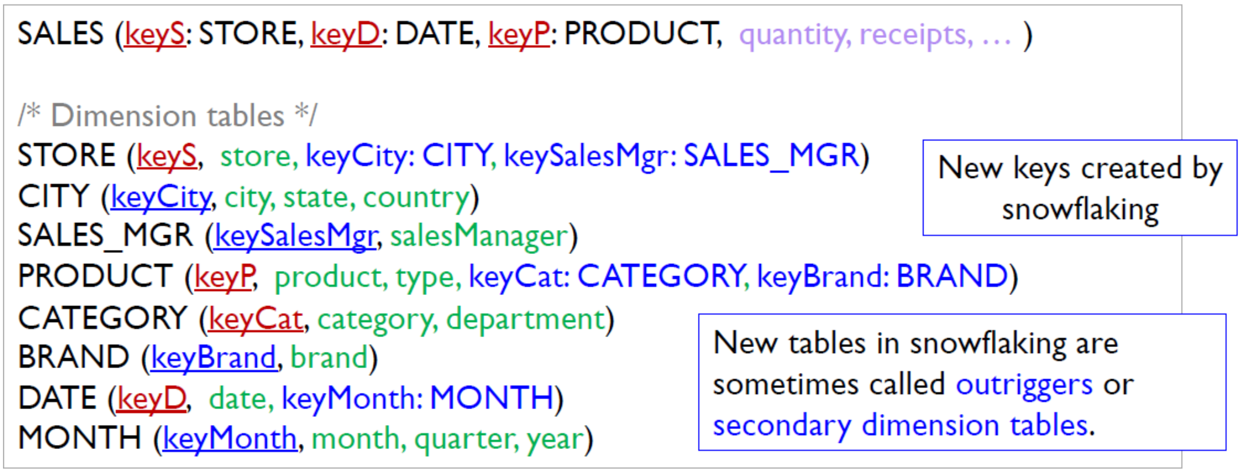
Snowflake schema

Snowflaking is a method of (partially) normalizing the dimension tables in a star schema.

- No need to do full normalization (e.g. 3NF)
- The result is known as snowflake schema

Break transitive functional dependency by moving some attributes to separate tables and relate these tables with new surrogate keys

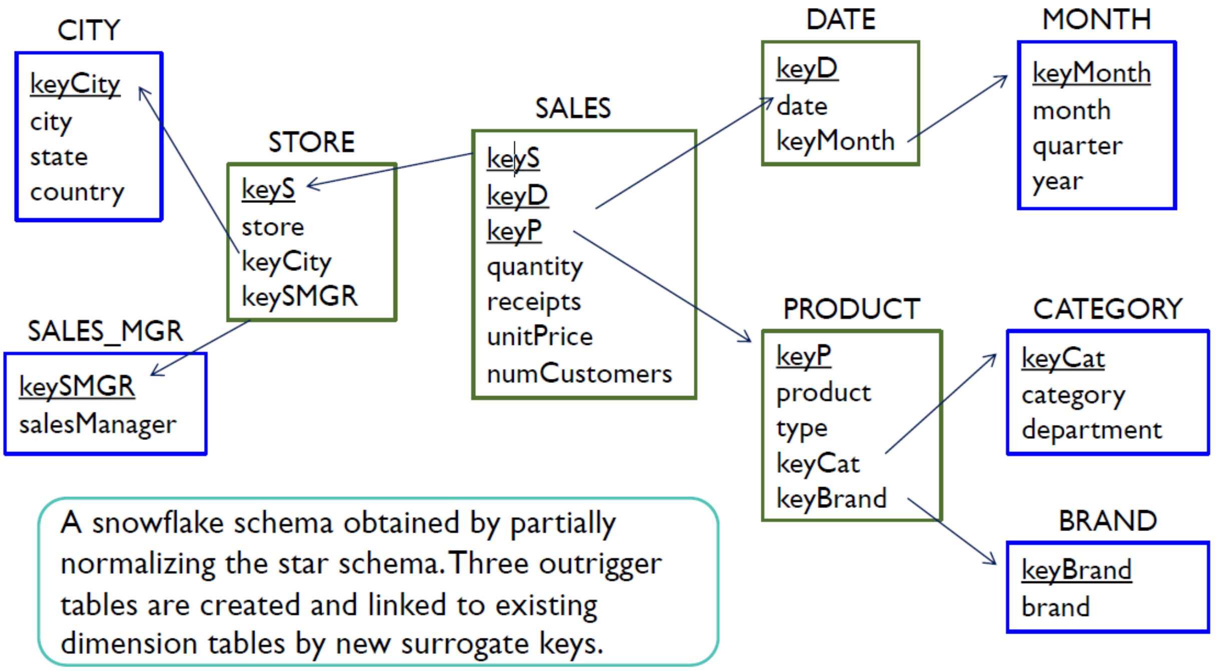



SALES (keyS: STORE, keyD: DATE, keyP: PRODUCT, quantity, receipts, …)

/* Dimension tables */
STORE (keyS, store, keyCity: CITY, keySalesMgr: SALES_MGR)
CITY (keyCity, city, state, country)
SALES_MGR (keySalesMgr, salesManager)
PRODUCT (keyP, product, type, keyCat: CATEGORY, keyBrand: BRAND)
CATEGORY (keyCat, category, department)
BRAND (keyBrand, brand)
DATE (keyD, date, keyMonth: MONTH)
MONTH (keyMonth, month, quarter, year)

New keys created by snowflaking

New tables in snowflaking are sometimes called outriggers or secondary dimension tables.



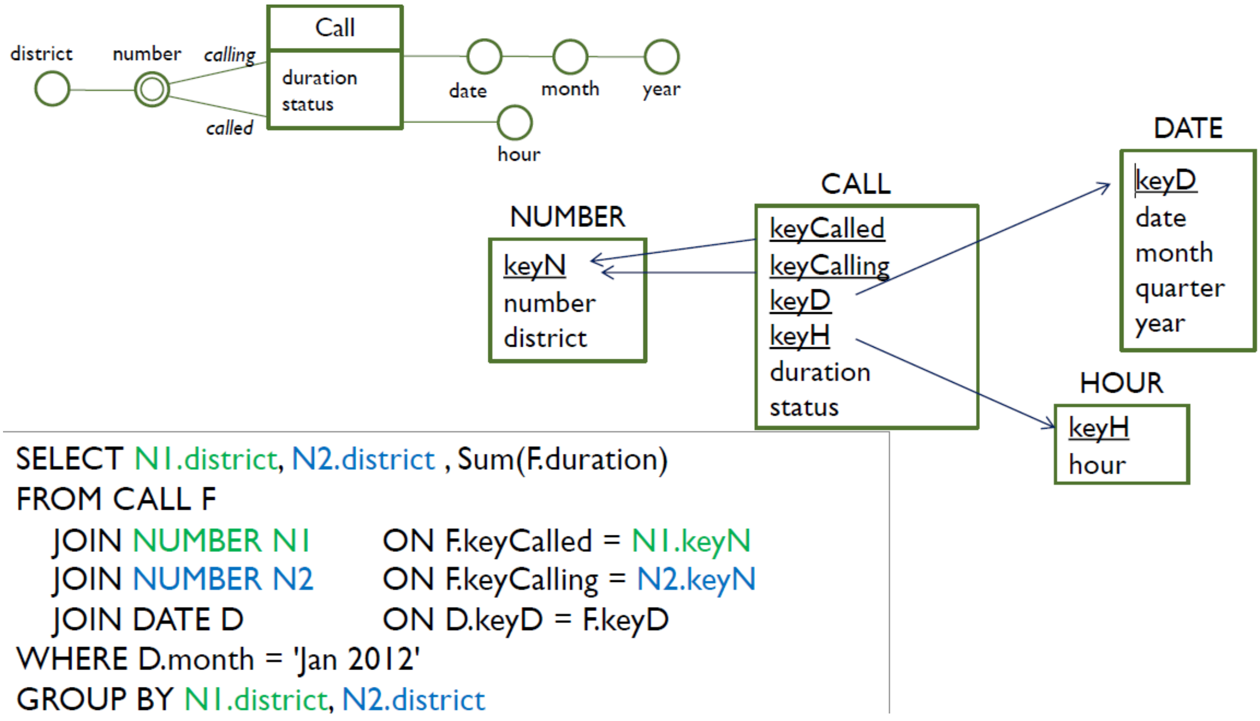
A snowflake schema obtained by partially normalizing the star schema. Three outrigger tables are created and linked to existing dimension tables by new surrogate keys.

Why Snowflake

- Some OLAP tools and reporting tools function better under the more normalized conditions of a snowflake design
- Separate a hierarchy from a very large dimension to save storage space
- Sharing a hierarchy among dimensions

Total sharing

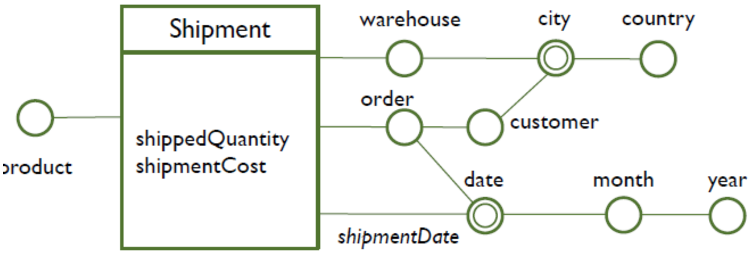
Two hierarchies contain exactly the same attributes used with different meanings

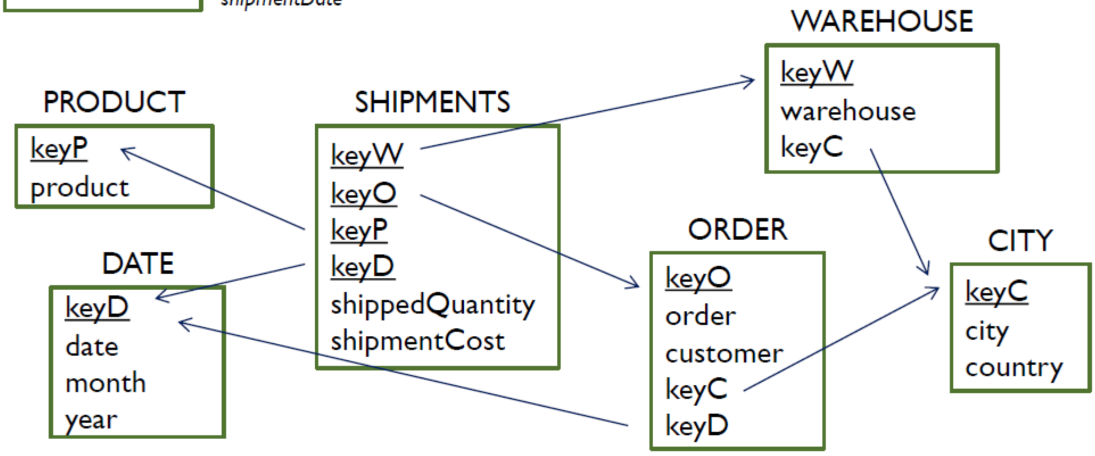


```
SELECT N1.district, N2.district , Sum(F.duration)
FROM CALL F
   JOIN NUMBER N1       ON F.keyCalled = N1.keyN
   JOIN NUMBER N2       ON F.keyCalling = N2.keyN
   JOIN DATE D          ON D.keyD = F.keyD
WHERE D.month = 'Jan 2012'
GROUP BY N1.district, N2.district
```

Partial sharing

Two hierarchies share some of their attributes




WAREHOUSE
keyW
warehouse
keyC

PRODUCT
keyP
product

SHIPMENTS
keyW
keyO
keyP
keyD
shippedQuantity
shipmentCost

ORDER
keyO
order
customer
keyC
keyD

CITY
keyC
city
country

DATE
keyD
date
month
year

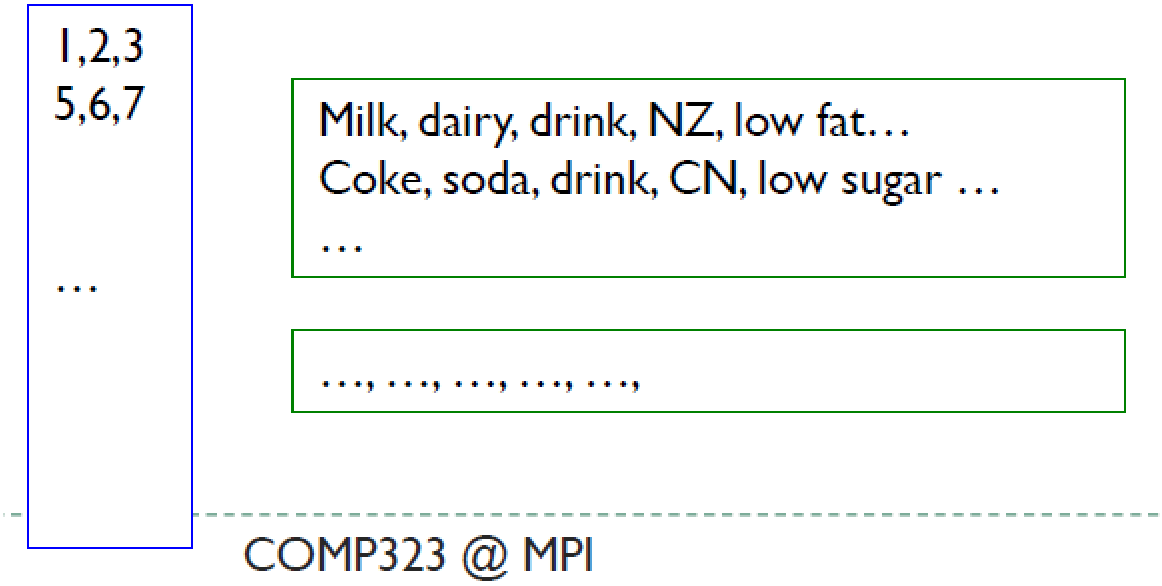
Fact vs Dimension tables

Fact table

- Deep, but not wide: it contains few attributes, but a large number of records.
- Most attributes are numerical measures that can be aggregated

Dimension table

- Wide, but not deep: it contains many attributes, but fewer records than the fact table
- Most attributes are textual attributes that are used to filter or sort data in fact table


1,2,3
5,6,7

…

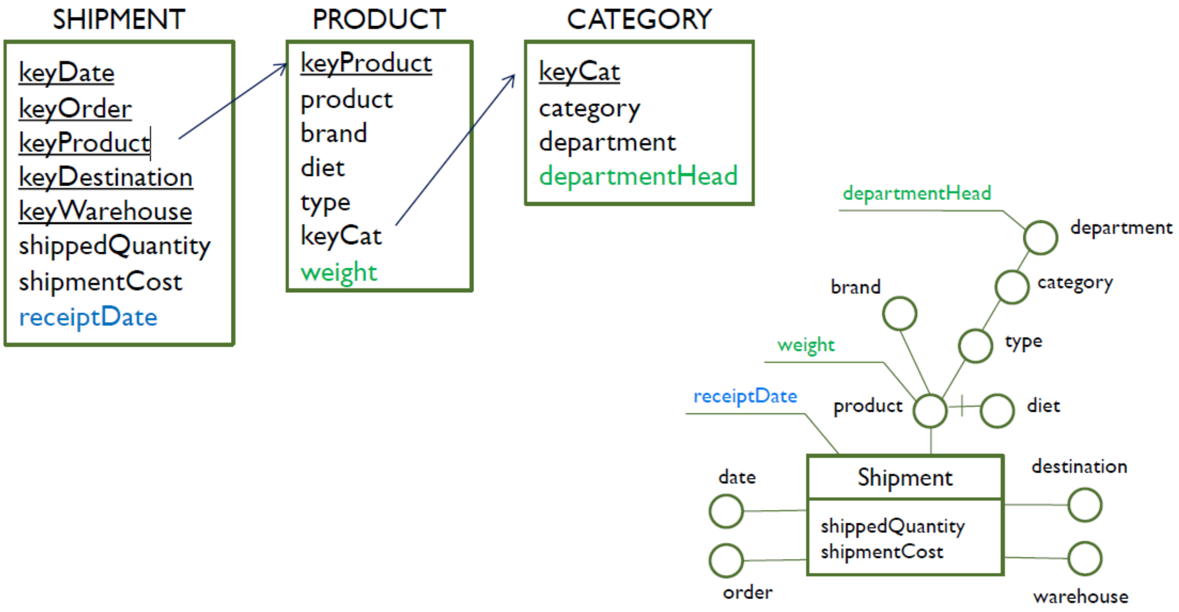
Milk, dairy, drink, NZ, low fat…
Coke, soda, drink, CN, low sugar …
…

…, …, …, …, …,

COMP323 @ MPI

Descriptive attributes

- A descriptive attribute contains information that cannot be used for aggregating, but that is considered useful to retain
- A descriptive attribute linked to a dimensional attribute must be included in the dimension table for the hierarchy that contains it.
- A descriptive attribute linked to a fact must be included in the fact table.



Degenerate(退化) dimensions

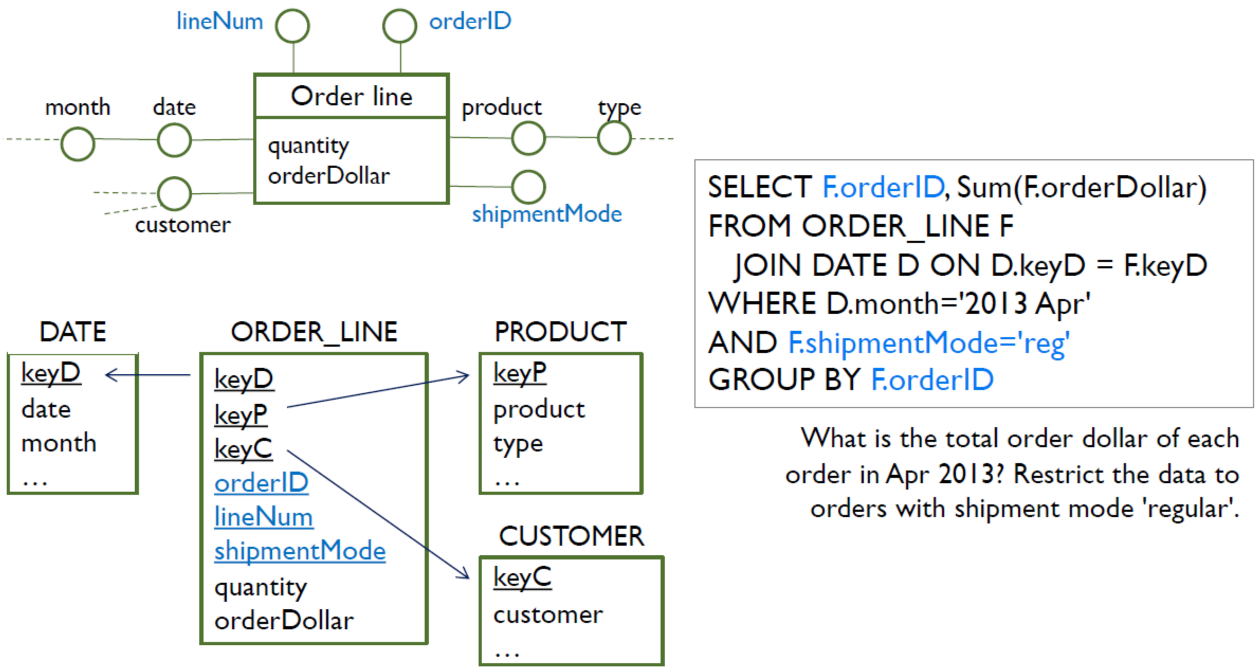
Some dimensions have only one attribute

If the size of the attribute is smaller than a surrogate key (e.g. <= 4 bytes), or if the cardinality of the attribute is very large (e.g. transaction ID),

- creating a separate dimension table waste storage space and requires a table join in query processing
- degenerate dimension is a better choice!

In degenerate dimension, we store the single attribute of the dimension directly in the fact table

- There is no foreign key and surrogate key for degenerate dimension
- This attribute is part of the composite primary key of the fact table
- Such degenerate dimension can still be used to filter and group data in fact



```
SELECT F.orderID, Sum(F.orderDollar)
FROM ORDER_LINE F
  JOIN DATE D ON D.keyD = F.keyD
WHERE D.month='2013 Apr'
AND F.shipmentMode='reg'
GROUP BY F.orderID
```

What is the total order dollar of each order in Apr 2013? Restrict the data to orders with shipment mode 'regular'.

Slowly-Changing Dimensions

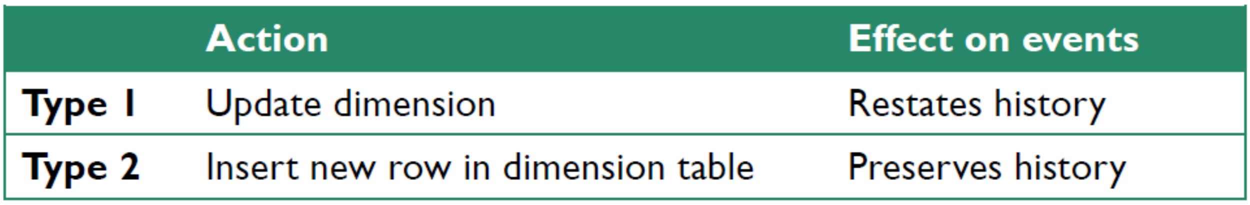
ETL process constantly inserts new rows to fact tables. Occasionally, it also needs to insert new rows to dimension tables
E.g. new products, new customers

In general, existing rows in fact tables do not change

Sometimes, changes in operational databases may result in changes in some dimensional attributes.

- E.g. correcting an error in the birthday of a customer
- E.g. a customer moves to a new address

The problem of how to update the dimension tables to reflect such changes is known as Slowly-Changing Dimensions



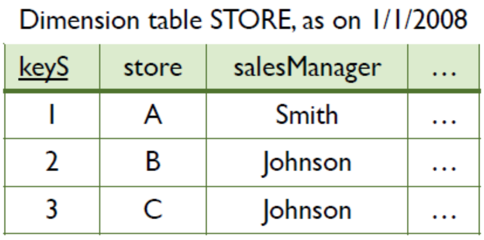
	Action	Effect on events
Type 1	Update dimension	Restates history
Type 2	Insert new row in dimension table	Preserves history

Type 1

Type 1 overwrites the attribute value in the dimension table row.

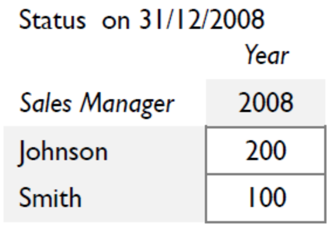
Preexisting events have a new context. All the events including past ones are always interpreted from the
viewpoint of the current value dimension attributes.

Used when no need to preserve the old value.
E.g. in case of correction of errors in operational systems



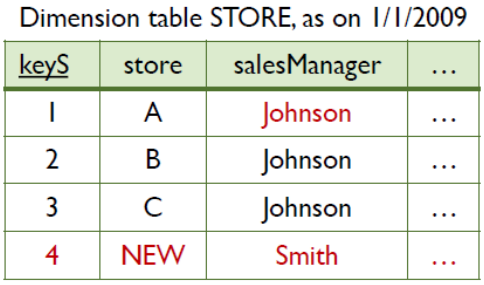
Dimension table STORE, as on 1/1/2008

keyS	store	salesManager	…
1	A	Smith	…
2	B	Johnson	…
3	C	Johnson	…



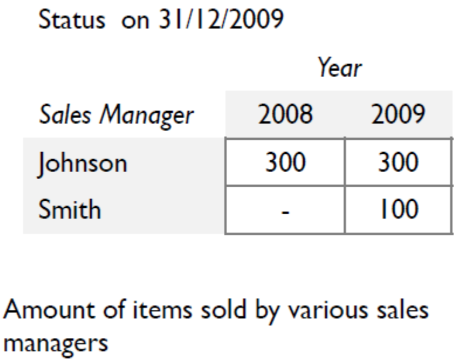
Status on 31/12/2008

	Year
Sales Manager	2008
Johnson	200
Smith	100



Dimension table STORE, as on 1/1/2009

keyS	store	salesManager	…
1	A	Johnson	…
2	B	Johnson	…
3	C	Johnson	…
4	NEW	Smith	…



Status on 31/12/2009

	Year	
Sales Manager	2008	2009
Johnson	300	300
Smith	-	100

Amount of items sold by various sales managers

Type 2

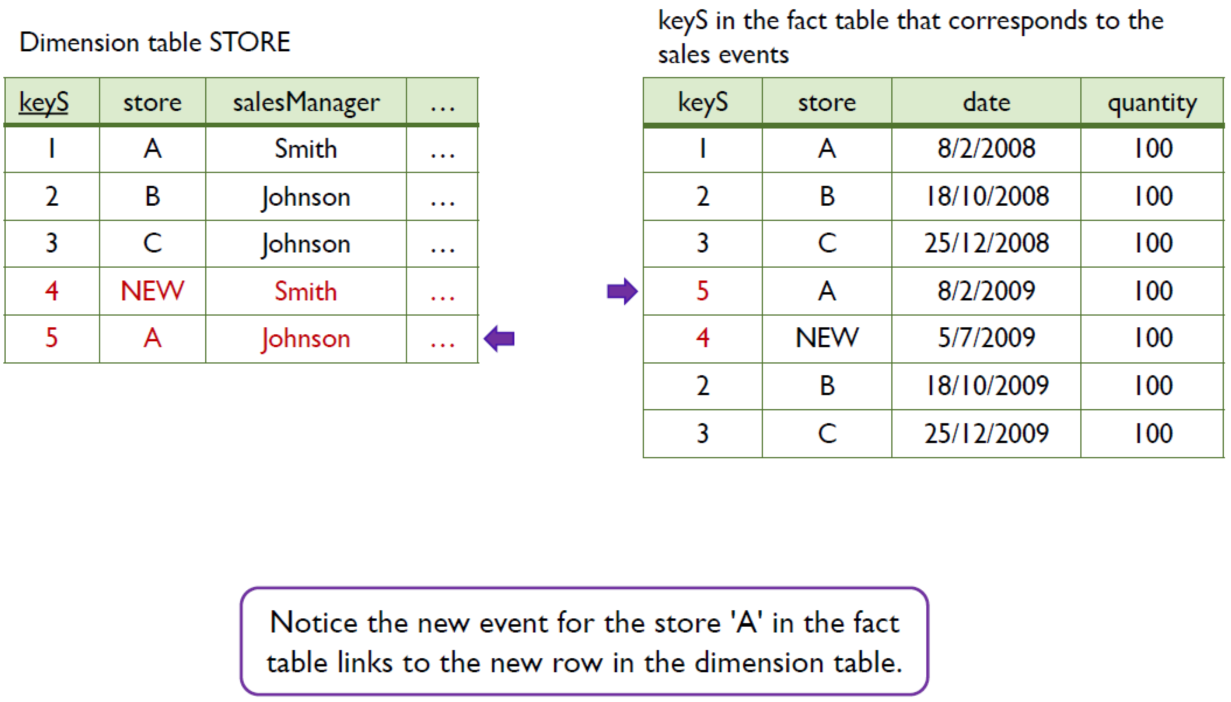
Type 2 change inserts a new row to store new values into the dimension table.

New events are associated with the new row while old events still link to the original row.

Historic context of events are preserved. An event is associated with the dimension row that was valid when the
event took place

May result in large dimension table in case of frequent changes.
Snowflaking may partially solve the problem

Surrogate keys support adding a new row that share the same natural keys for two versions of a dimension instance



Dimension table STORE

keyS	store	salesManager	…
1	A	Smith	…
2	B	Johnson	…
3	C	Johnson	…
4	NEW	Smith	…
5	A	Johnson	…

keyS in the fact table that corresponds to the sales events

keyS	store	date	quantity
1	A	8/2/2008	100
2	B	18/10/2008	100
3	C	25/12/2008	100
5	A	8/2/2009	100
4	NEW	5/7/2009	100
2	B	18/10/2009	100
3	C	25/12/2009	100

Notice the new event for the store 'A' in the fact table links to the new row in the dimension table.