Presentation: Dimensional Modelling 3

Erik Perjons

DSV, Stockholm University

More about facts and fact tables

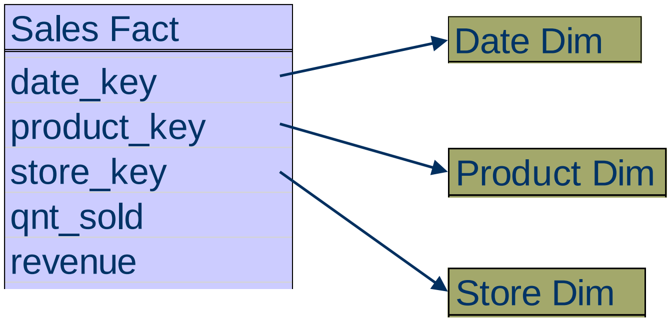
Type of Facts

- Additive Facts
- Semi-Additive Facts
- Non-Additive Facts

Additive Facts

- Additive Facts – are facts that are additive across **all** dimensions
- Example: Sales amount, Cost dollar amount, Sales quantity

Additive Facts



Aggregate on date

28/3, paper1, store1, 15, 150
29/3, paper1, store1, 35, 350

50, 500

OK to aggregate the facts quantity sold (qnt_sold) and revenue

28/3, paper1, store1, 25, 250
28/3, paper1, store2, 45, 450

70, 700

OK to aggregate the facts quantity sold (qnt_sold) and revenue

Aggregate on store

28/3, paper1, store1, 25, 250
28/3, paper2, store1, 10, 150

35, 350

Aggregate on store

OK to aggregate the facts quantity sold (qnt_sold) and revenue

Semi-Additive Facts

- Semi-Additive Facts – are facts that are additive across **some** dimensions
- Example: Account balance, Inventory level
- Often not additive using the Date dimension

Semi-Additive Facts

Sales Fact
date_key
product_key
store_key
qnt_sold
revenue
customer_count

Date Dim

Product Dim

Store Dim

Aggregate on product

28/3, paper1, store1, 25, 250, 20
28/3, paper2, store1, 35, 350, 30

60, 600, 50

NB! customer_count is not additive across the product dimension

28/3, paper, store1, 15, 150, 10
29/3, paper, store1, 35, 350, 30

50, 500, 40

Aggregate on date

OK to aggregate the facts quantity sold (qnt_sold), revenue, and ***customer_count***

28/3, paper, store1, 25, 250, 20
28/3, paper, store2, 45, 450, 40

70, 700, 60

Aggregate on store

OK to aggregate the facts quantity sold (qnt_sold), revenue, and ***customer_count***

Is the number of customers who bought either paper towels or tissue paper 50?

Customer_count is a semi-additive fact in this case

No, the number could be anywhere between 30 and 50.

Semi-Additive Facts

- All measures that record a static level, such as account balance and inventory level, are non-additive across time.
- However, these measures may be usefully aggregated across time by averaging over the number of time periods.

Non-Additive Facts

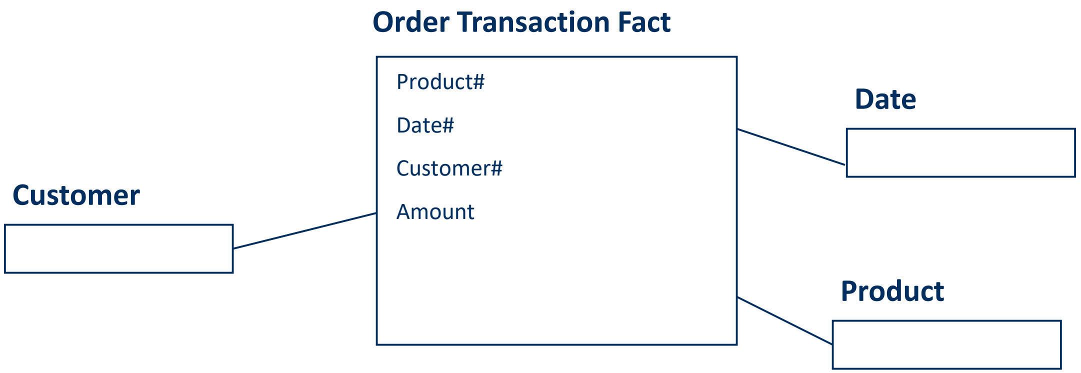
- Non-Additive Facts – are facts that cannot be added at all, i.e., not be added along any dimension
- Example: percentages, ratios, unit price, temperature, blood pressure
- Yan still do some form of calculations on these facts, for example, apply median or average

Type of Fact Tables

- Transaction Fact Table
- Periodic Snapshot Fact Table
- Accumulating Snapshot Fact Table

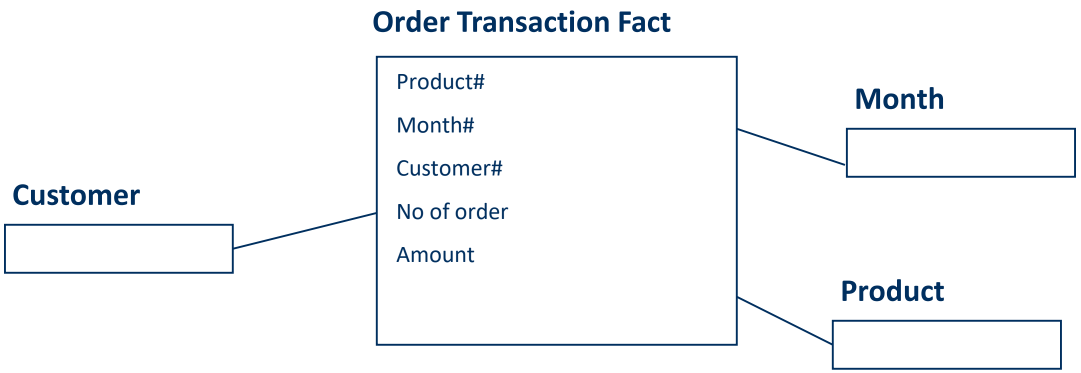
Transaction fact tables

- Transaction fact tables represent an event that occurred at an instantaneous point in time
- A row exist in the fact table for a given customer or product only if the transaction event has occurred



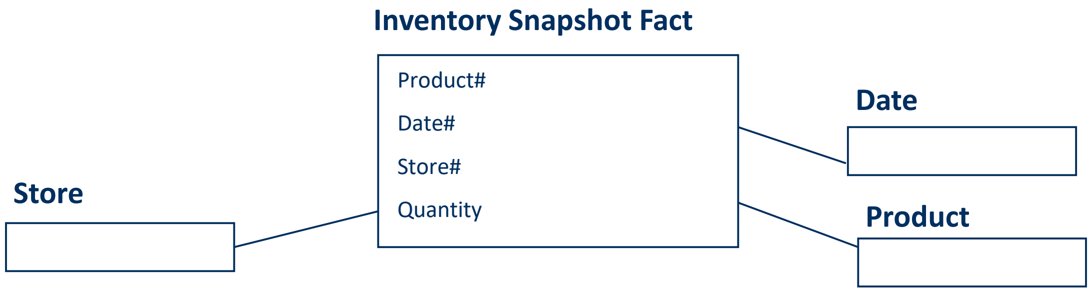
Transaction fact tables

- Transaction fact table may also have been aggregated on date, for example all transaction for a day, week, month – and is still called a transaction fact table



Periodic snapshot fact tables

- Periodic snapshot fact table – shows a picture/state of, for example, the quantity of products in different stores' inventories, at an end of a day, week, or month, then another picture in the end of next period, and so on.
- The periodic snapshots are stacked consecutively into the fact table



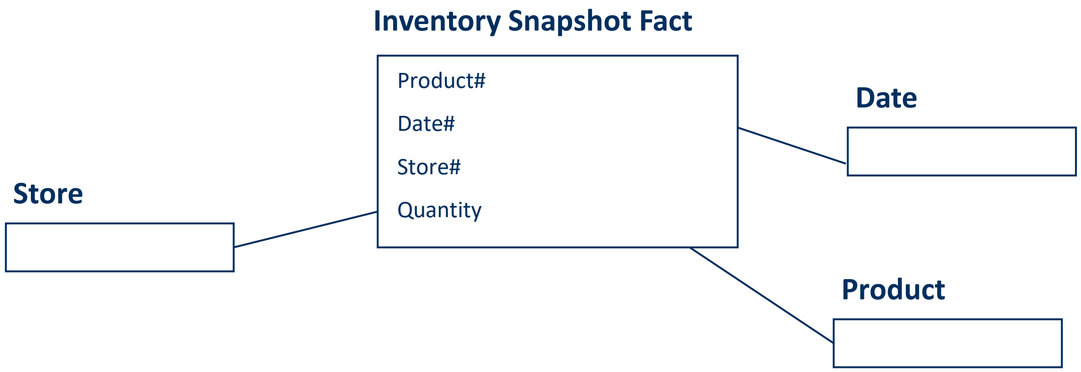
Periodic snapshot fact tables

- Periodic snapshot fact table represents a snapshot of data (facts) at specific point in time

Store inventory periodic snapshot schema

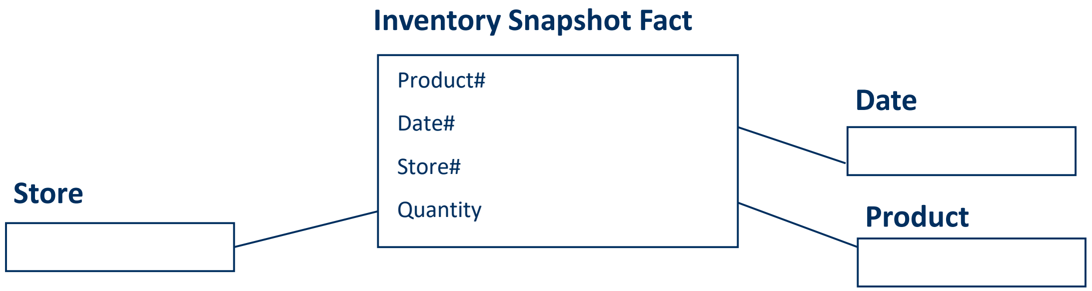
Periodic snapshot fact tables

- Periodic snapshot fact table is often the only place to easily retrieve a regular, predictable, trendable view of on some key business performance metrics



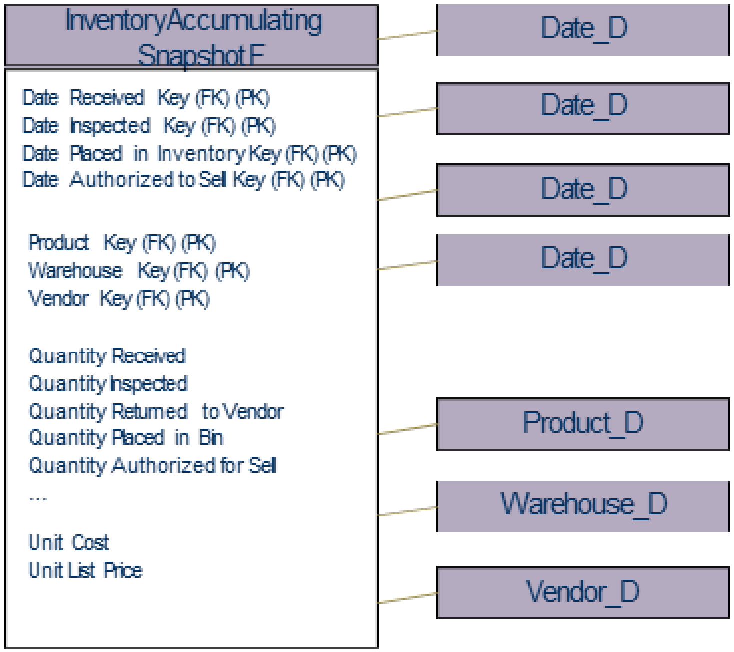
Periodic snapshot fact tables

- All measures that record a static level, such as account balance and inventory level, are **non-additive across time**, but note, they may be **semi-additive** using other dimensions
- However, these measures may be usefully **aggregated across time by averaging over the number of time periods**.



Accumulating snapshot fact tables

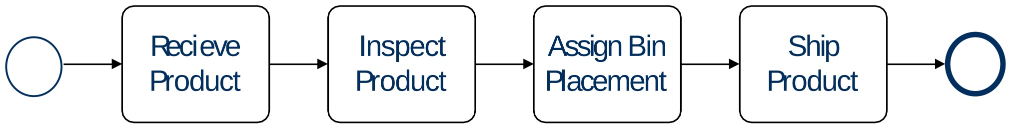
- Accumulating snapshot fact table - represents an indeterminate time span, covering a the complete life of a transaction
- Almost always the fact tables have multiple time/date stamps, representing the predictable major events or phases that take place during the course of lifetime



Accumulating snapshot fact tables

Stockholms universitet

- Accumulating snapshot fact tables are used for processes that have a definite beginning, definite end, and identifiable milestones in between



Accumulating snapshot fact tables

Stockholms universitet

- In sharp contrast to the other fact table types, we purposely revisit accumulation snapshot fact table rows TO UPDATE (!!!) them. That is, we revisit them as more information becomes available
- Since many of these dates are not known when the fact row is loaded, we must use surrogate date key to handle undefined dates
- There need to be a row in the date dimension with the date=“unknown” or “to be determined”, when we first load the row in the fact table

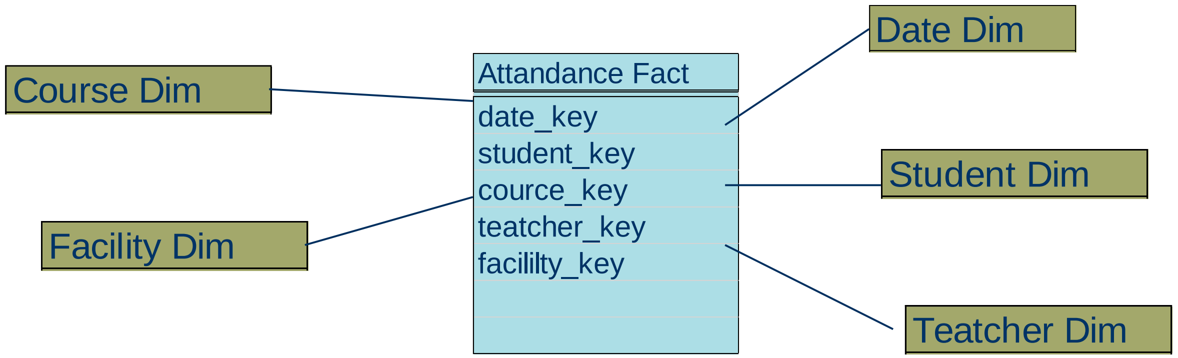
Factless fact tables

Stockholms universitet

- Some fact tables quite simply have no measured facts
- These fact tables are useful to describe events and coverage, i.e. the tables contain information that something has (event tracking) or has not (coverage table) happened
- There are several types of factless fact tables, two of the most common are:
  - event tracking tables
  - coverage tables

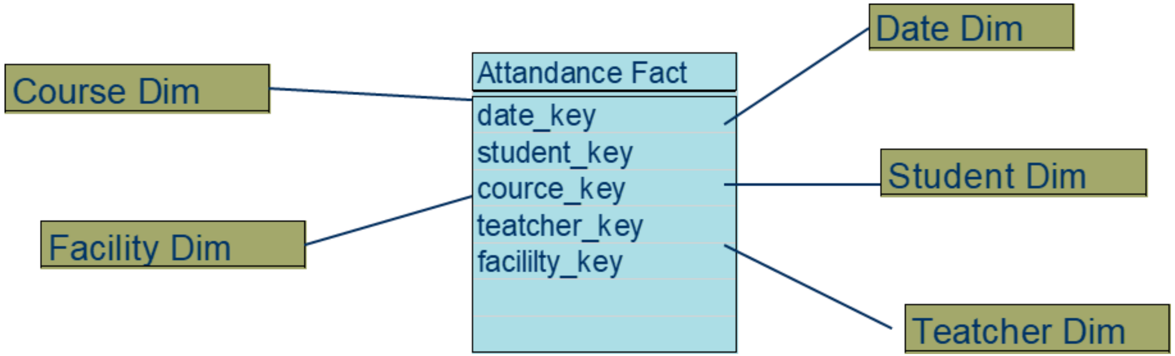
Event tracking (factless fact) tables

- An event tracking table - records events, e.g. records every time a student attends a course (see figure), or people involved in accidents and vehicles involved in accidents



Event tracking (factless fact) tables

- An event tracking table - contains a concatenated key that represent a focal event which is identified by the combination of conditions referenced in the dimension tables



Other types of factless fact tables

Another problem that can be addressed by factless fact tables

- Many to many relationships (M-to-M) between entities (tables) are difficult to deal with in any database design situation. For example, a customer can have many accounts and an account may belong to many customers
- A factless fact table can be created to capture the relationship between the tables

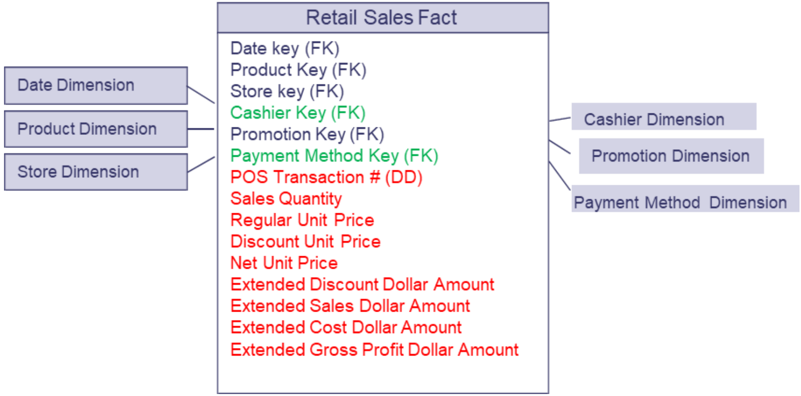
Coverage (factless fact) tables

- How we can answer questions for which there is no event in the business process?
- We can store all possibilities in a factless fact table in form of a coverage fact table

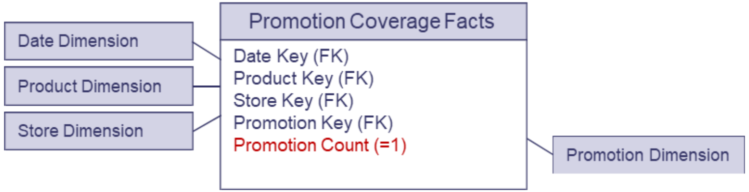
Coverage (factless fact) table

- An example: What products were on promotion but did not sell?
  - The sales fact table records only the SKUs actually sold.
  - Therefore, we need to create a factless fact table that cover all product that is part of the promotion

Occurred Events



Promotion coverage factless fact table



More about dimensions

Slowly Changing Dimensions

Problem to solve:

Dimension attribute values change over time, e.g., a product that belong to a department or product category, later belong to another department or category

The assumption:

The key does not change, but some of the attribute values does.

Slowly Changing Dimensions

- Type 1: Overwrite the dimension record (attribute value) with the new values, thereby losing history
- Type 2: Create a new additional dimension record using a new value of the surrogate key
- Type 3: Create a new field in the dimension record to store the new value of the attribute

Type 1

Stockholms universitet

Overwrite the old value of an attribute with a new one

e.g.

12334	Mary	Jones		~~single~~ married

Product Key	Description	Department	SKUNumber (NK)
12345	IntelliKidz 1.0	~~Education~~ Strategy	ABC922-Z

Change name of a department

+ easy to implement

+ OK, if there is no use in keeping the old value (e.g., mobile number)

- avoids the real goal of data warehousing, which is to accurately track history

- any pre-aggregates based on the attribute values for married (singe/married) need to be rebuilt

Type 2

- Create a new additional dimension record
- The predominant technique for handling slowly changing dimensions
- A generalised (surrogate) key is required (which is a responsibility of the data warehouse team)

PrimaryKey	Product description	Department	SKU Number (Natural key)
12345	IntelliKidz 1.0	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	ABC922-Z

Type 2

Stockholms universitet

- We can use/constrain the attribute Production description (“IntelliKidz 1.0”) or SKU number (“ABC922-Z”) and the query will automatically fetch both IntelliKidz product dimension rows and join the fact table for the complete product history

PrimaryKey	Product description	Department	SKU Number (Natural key)
12345	IntelliKidz 1.0	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	ABC922-Z

Type 2

Stockholms universitet

- We can add also add additional attributes:
  - Row Effective Date,
  - Row Expiration Date (default: December 31, 9999)
  - Current Row Indicator

Product Dimension
Product Key (PK)
SKU Number (Natural Key)
Product Description
Department Name
...
Row Effective Date
Row Expiration Date
Current Row Indicator

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	January 1, 2012	December 31, 9999	Current

Rows in Product dimension following department reassignment:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	January 1, 2012	January 31, 2013	Expired
25984	ABC922-Z	IntelliKidz	Strategy	...	February 1, 2013	December 31, 9999	Current

Type 2

+ history is stored

+ can track as many dimensional attribute value changes as required

+ no need to rebuilt pre-aggregations

- could lead to an accelerated dimensional table growth of rows.

Type 2

- Another solution Use of smart keys using extra digits in the end of the key. Recommended by Kimball 1996

Fact table

		…	
		12334001	
		12334001	
		…	
		12334001	
		12334002	
		…	
		12334002	
		…	

Dimension table

…				
12334001	Mary	Jones		single
…				
12334002	Mary	Jones		married
…				

Type 3

- Create a new field in the dimension record

Nr	First Name	Family Name	Original / Previous Marrital Status	Current Marrital Status	Effective Date
12334	Mary	Jones	single	married	15/6 1987

+ Allow us to see new and historical fact data by either the new and prior attribute values. Enable alternate reality, i.e., see two views of the world simultaneously

- What if an attribute change values several times?

Rapidly changing dimensions

- What if the changes are fast?
- Break off some of the attributes into their own separate dimension(s), a minidimension(s).
  - force the attributes selected to the minidimension to have relatively small number of discrete values
  - build upp the minidimension with all possible discrete attributes combinations
  - construct a surrogate key for this dimension

Minidimension

Stockholms universitet

Demographics dim.
demog_key
income_band
education_level
marrital_status

Three values

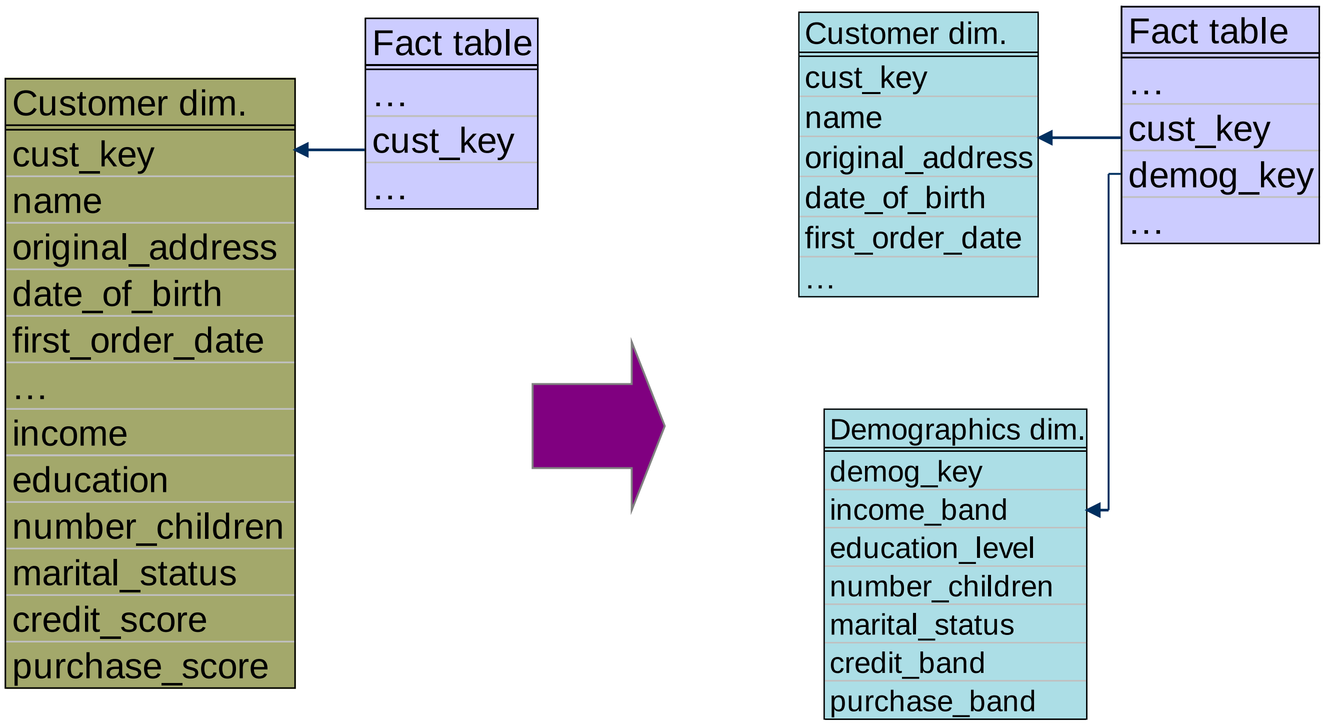
Two values

Two values

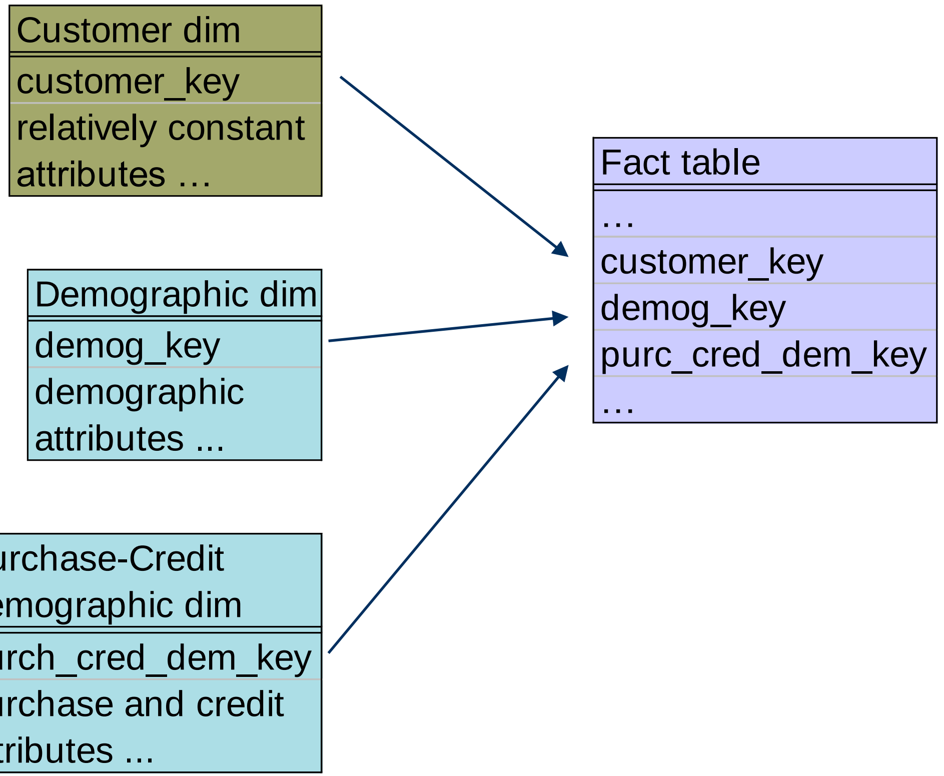
3*2*2=12 rows

D1	-100 000	Graduate	Married
D2	100 000-200 000	Graduate	Married
D3	200 000-	Graduate	Married
D4	-100 000	Non-graduate	Married
D5	100 000-200 000	Non-graduate	Married
D6	200 000-	Non-graduate	Married
	..cont	*..cont*	*..cont*

Demographic Minidimension



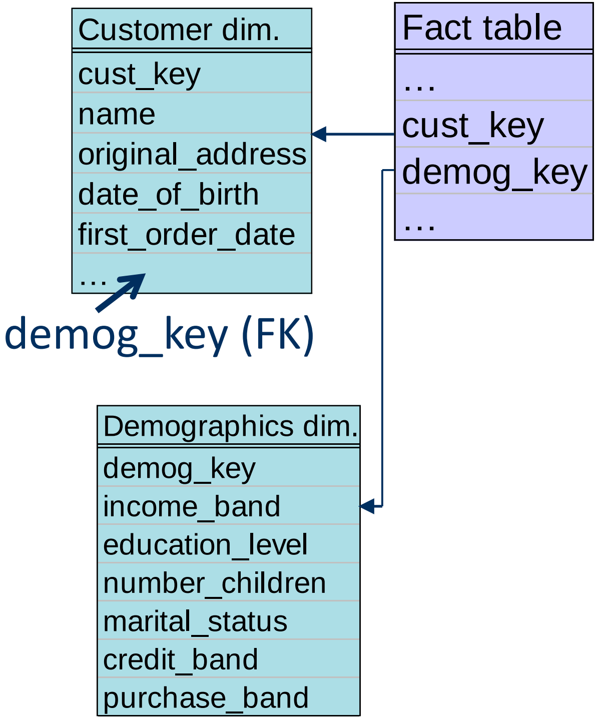
Two Minidimensions



Using Minidimension

- **Advantages**
  - frequent ‘snapshoting’ of customers profiles with no increase in data storage or data complexity
- **Drawbacks**
  - the demographic attributes are clumped into banded ranges of discrete values – and it is impractical to change the set of value bands at a later time
  - the demographic dimension itself can not be allowed to grow too large

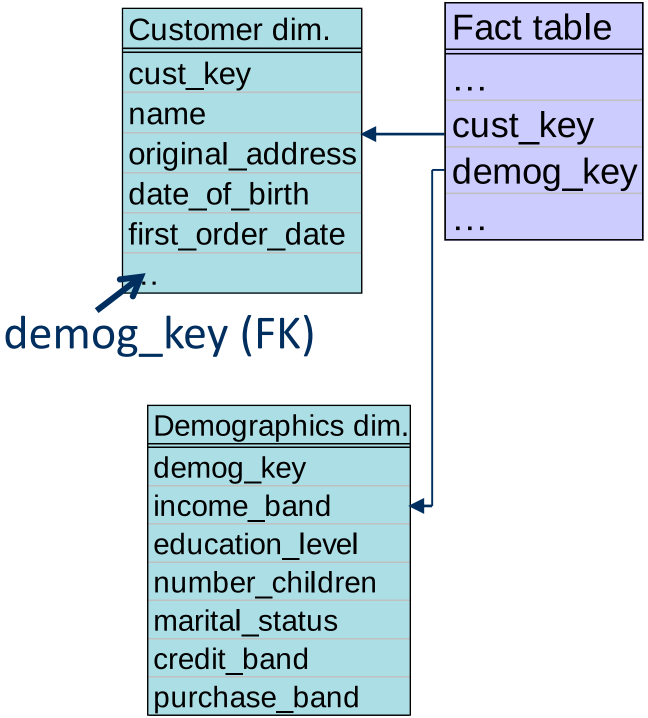
Another problem with minidimensions



- If a customer are not involed in any transaction, there is no information about this customer demographic state in this star-join schema
- Solutions: add a demograhic key as a foreign key in the customer dimension

Minidimension vs Outriggers

Stockholms universitet



Minidimension = If the demograhic key is part of the fact table composite key

Outrigger = if the demograhic key is a foreign key in the customer dimension

Another example of minidimension

Stockholms universitet

Demographics Key	Age Band	Purchase Frequency Score	Income Level
1	21-25	Low	<$30,000
2	21-25	Medium	<$30,000
3	21-25	High	<$30,000
4	21-25	Low	$30,000-39,999
5	21-25	Medium	$30,000-39,999
6	21-25	High	$30,000-39,999
…	…	…	…
142	26-30	Low	<$30,000
143	26-30	Medium	<$30,000
144	26-30	High	<$30,000
…	…	…	…

Date Key	Customer Key	Demographics Key	…	…2
20160119	1	1	…	…
20160518	1	2	…	…

Customer Dimension

Customer Key (PK)
Customer ID (NK)
Customer Name
Customer Address
Customer City-State
Customer State
Customer ZIP-Postal Code
Customer Date of Birth

Fact Table

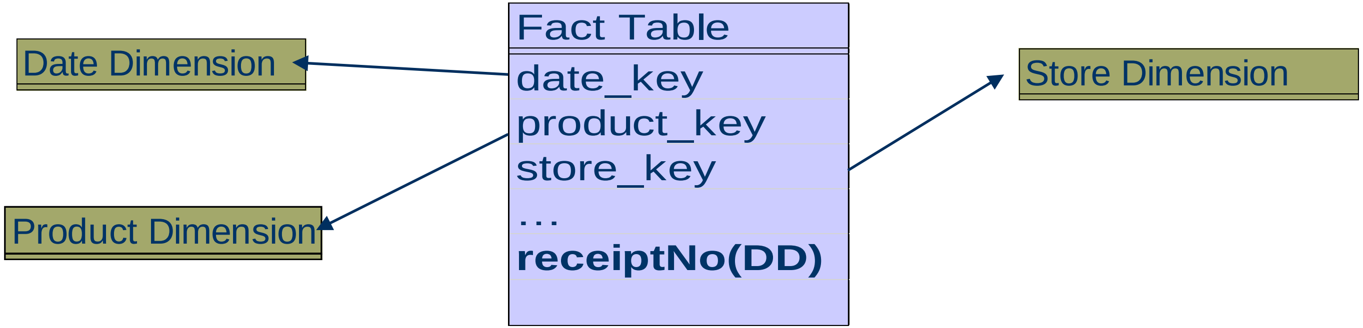
Date Key (FK)
Customer Key (FK)
Demographics Key (FK)
More FKs…
Facts…

Demographics Dimension

Demographics Key (PK)
Age Band
Purchase Frequency Score
Income Level

Degenerate Dimension

- A degenerate dimension is represented by a dimension key attribute(s) with no corresponding dimension table
- Often transaction number, receipt number, etc



Junk Dimension

- A **junk dimension** - is a convenient grouping of attributes and flags into a useful dimensional framework to get them out of a fact table or to avoid adding a number of extra dimensions into a useful dimensional framework

Junk Dimension

- When a number of miscellaneous text attributes or flags exist, the following design alternatives should be avoided:
  - Leaving the flags and attributes unchanged in the fact table record (the fact table will become large)
  - Making each flag and attribute into its own separate dimension (the fact table will become large)
  - Stripping out all of these flags and attributes from the design (missing info/constrain alternatives)

“Combined” dimensions

Stockholms universitet

Problem to address: Different services could be used in the same time

Solution1 : Create a new dimension for each service – and they could be combined in the fact table (however: many foreign keys in the fact table)

Solution2 : Create a new row for each service (however: problems with aggregation on the fly)

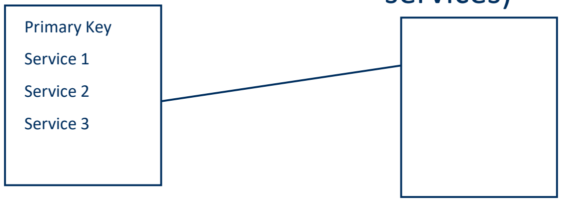
Solution3 : Create a dimension consisting of all service combinations, which means that there is a row/instance for each combination in the dimension tables (compare the minidimension solution)

Solution 3

PK	Service 1	Service 2	Service 3
1	Y	N	N
2	N	Y	N
3	N	N	Y
4	Y	Y	N
5	Y	N	Y
6	N	Y	Y
7	Y	Y	Y

Service dim

Transaction Fact Table – (including used services)



Heterogenous products

Heterogeneous Products

Problem to address:

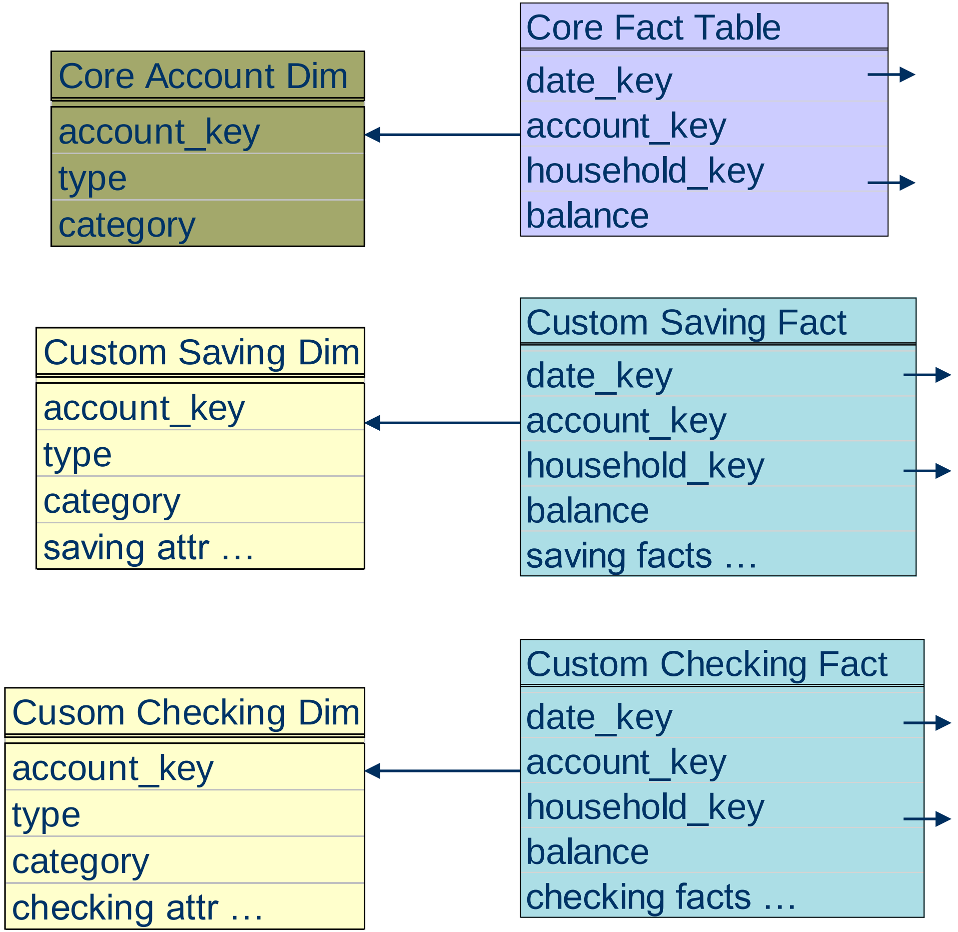
- Some products have many distinguishing attributes and many possible permutations (usually on the basis of some customised offer).
- This results in immense product dimensions and bad browsing performance

Heterogeneous Products

Solution:

- In order to deal with this, fact tables with accompanying product dimensions can be created for each product type - these are known as ***custom fact tables***
- Primary and common core facts on the products types are kept in a ***core fact table*** (but can also be copied to the conformed fact tables)

Heterogeneous Products



Aggregrations

Aggregations

- Aggregations can be created *on-the-fly* or by the process of *pre-aggregation*
- An aggregate is a **fact table record** representing a summarisation of base-level fact table records
  - Category-level product aggregates by store by day
  - District-level store aggregates by product by day
  - Monthly sales aggregates by product by store
  - Category-level product aggregates by store district by day
  - Category-level product aggregates by store district by month

New Tables for Aggregates

Category Aggregate Sales Fact
date_key
category_key
store_key
promo_key
dollars_sold
units_sold
dollars_cost
customer_count

Category Dimension
categry_key
category
department
…

New Tables for Aggregates

P_Key	Produc name	Subcategory	Category	LEVEL
P11	white napkin	napkin	paper	base
P12	pink napkin	napkin	paper	base
P13	red napkin	napkin	paper	base
P24	Eko tissue	tissue	paper	base
P25	Leni tissue	tissue	paper	base

SK	Subcat	Category
P10	napkin	paper
P20	tissue	paper

CK	Category
P100	paper

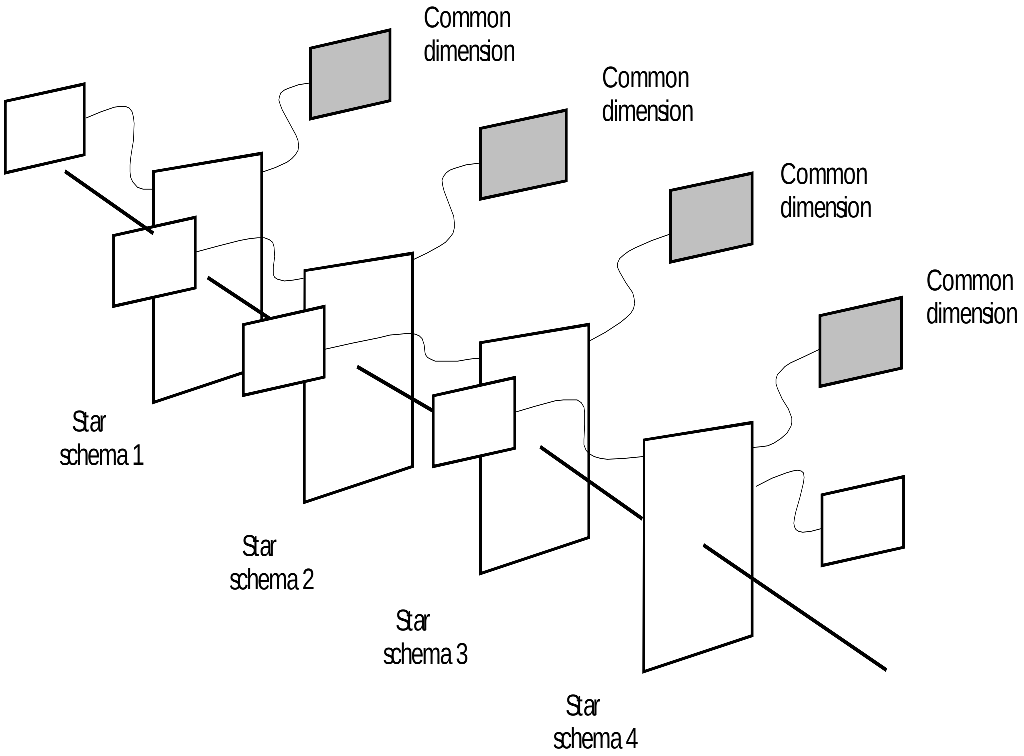
Date_key	P_Key	$ sold
1-May	P12	100
1-May	P11	200
1-May	P25	300
2-May	P12	250
3-May	P12	100
4-May	P13	50
4-May	P24	150
1-May	P10	300

Date_key	SK	$ sold
1-May	P20	300
2-May	P10	250
3-May	P10	100
4-May	P10	50
4-May	P20	150

Date_key	CK	$ sold
1-May	P100	600
2-May	P100	250
3-May	P100	100
4-May	P100	200

Family of stars again

A family of stars



A family of stars

- A dimensional model of a data warehouse for a large data warehouse consists of between 10 and 25 similar-looking star-join schemas. Each star join will have 5 to 15 dimensional tables
- Conformed (shared) dimensions for drill-across

Conformed dimensions and facts

- **Conformed dimensions** – has consistent dimension keys, consistent attribute column names, consistent attribute definitions and consistent attribute values. This make drill-across possible from one fact table to another via the conformed dimension

- **Conformed facts** – means conformed fact definition, i.e., definitions of revenue, profit, standard costs, measures of quality and customer satisfaction.

- Note: If it is impossible to conform a fact exactly, then you should give different name to the different interpretation