Lecture 2 and 3 - Dimensional Modelling

Reading Directions

L2 [K&R] chapters 2-8

L3 [K&R] chapters 9-13, 15

Keywords

facts, attributes, dimensions, granularity, dimensional modeling, time, semi-additive facts, dense fact tables, sparsity, skinny fact tables, keys, slowly changing dimension, rapidly changing dimensions, large dimensions, demographic minidimension, degenerate dimension, junk dimension, heterogeneous products, many-to-many relationships, factless fact table, bridge table, family of stars, stove pipe problem, data warehouse bus, value chains, the design process, aggregates, sparcity failure, aggregation navigator, bitmap indexing, extended SQL, ROLAP and MOLAP servers

Some basic concepts

- Fact
  - "something not known in advance",
  - an observation
  - many facts (but not all) have <u>numerical</u>, <u>continuously</u> values

    e.g., the price of a product, quantity

- Attribute
  - "describe a characteristic of a tangible thing"
  - "we do not measure them, we usually know them"
  - usually <u>text</u> fields, with <u>discrete</u> values

    e.g., the flavour of a product, the size of a product

Some basic concepts 2

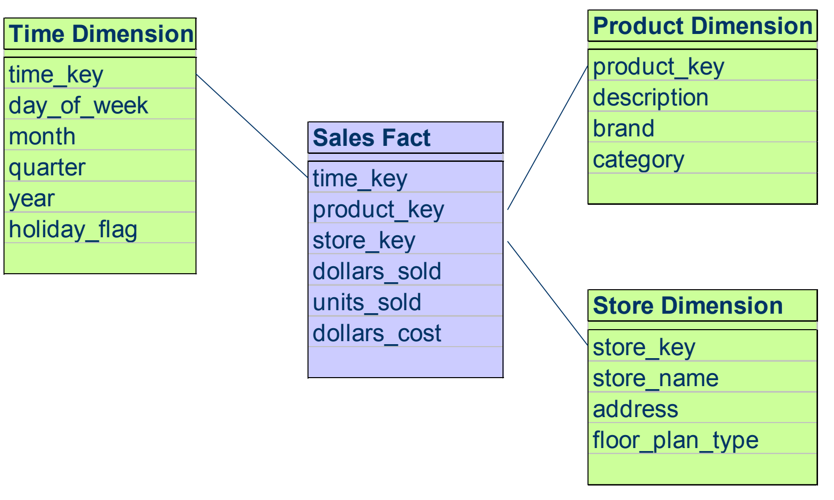
- Dimension
  - a business perspective from which data is looked upon
  - "a collection of text like attributes that are highly correlated"

    e.g. Product, Store, Time

- Granularity
  - the level of detail of data contained in the data warehouse

    e.g. Daily item totals by product, by store

Example of a Dimensional Model



The Standard Template Query

```
SELECT p.brand, sum(f.dollar), sum(f.units)
FROM salesfact f, product p, time t
WHERE f.prductkey = p.productkey
    AND f.timekey = t.timekey
    AND t.quarter = '1Q1995'
GROUP BY p.brand
ORDER BY p.brand
```

An Example Answer Set

Row header

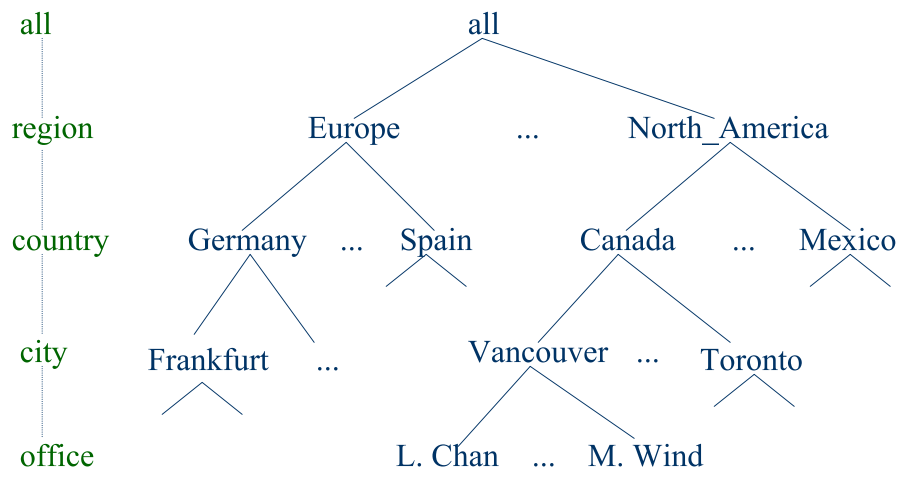
Brand	Dollar Sales	Unit Sales
Axon	780	263
Framis	1004	509
Widget	213	444
Zapper	95	39

Aggregated fact

The Time Dimension

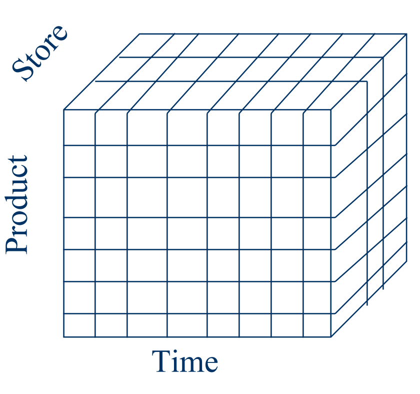
Time Dimension
time_key
day_of_week
day_nr_in_month
day_nr_overall
week_nr_in_year
week_nr_overall
month
month_nr_overall
quarter
fiscal_period
holiday_flag
last_day_in_month_flag
season
event

The Concept of Hierarchy

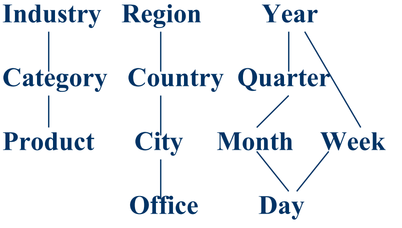


Multidimensional Data

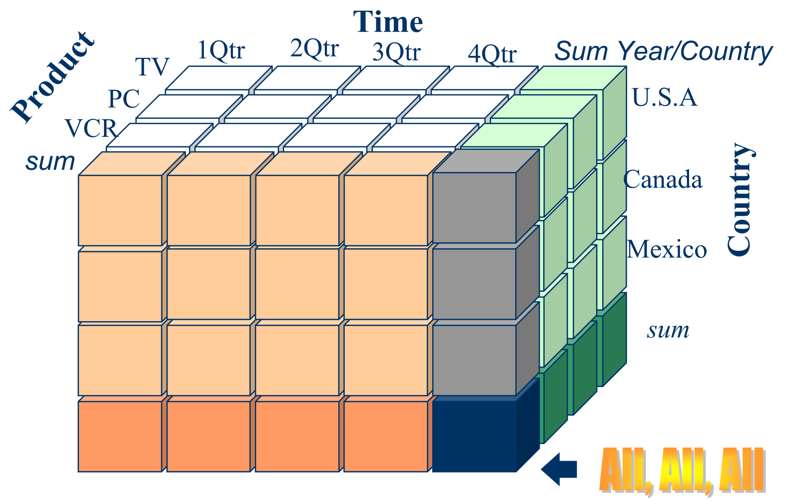
- Sales volume as a function of product, month, and region



Dimensions: Product, Location, Time
Hierarchical summarization paths



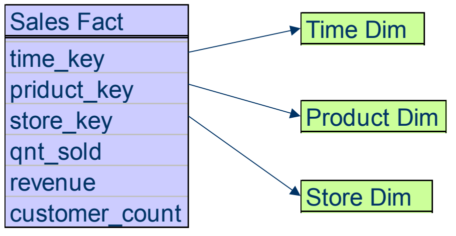
A Sample Data Cube



Facts

- (Perfectly) Additive
  - a fact is additive if it make sense to add it across <u>all</u> the dimensions
    - e.g., discrete numerical measures of activity, i.e., quantity sold, dollars soled
- Semiadditive
  - a fact is semiadditive if it make sense to add it along <u>some</u> of the dimensions only
    - e.g., numerical measures of intensity, i.e., account balance, inventory level
- Non-additive
  - facts that can not be added at all
    - e.g., measurement of room temperature

Facts and the Additive Property



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

50, 500, 40

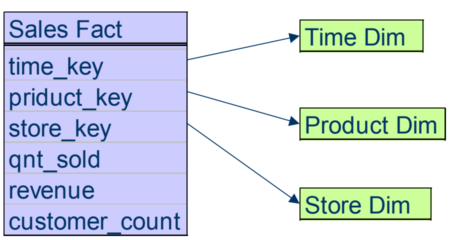
28/3, paper, store1, 25, 250, 20
28/3, paper, store2, 45, 450, 40

70, 700, 60

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

60, 600

Semiadditive fact - example



NB! customer_count is not additive across the product dimension

28/3, tissue paper, store1, 25, 250, 20
28/3, paper towels, store1, 35, 350, 30

50

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

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

Numerical Measures of Intensity

- 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.
- Note that, the SQL AVG can not be used for this.
  - What is the average daily inventory of a brand in a geographic region during a given week?
  - Let the brand cluster 3 products, the region has 4 stores, and we have 7 days/week.
  - Using the SQL AVG would divide the summed value into 3*4*7=84
  - The correct answer is to divide the summed inventory value by 7

Skinny fact tables

- As the fact table contains the vast volume of records it is important that it is memory space efficient
- Foreign keys are usually represented in integer form and do not require much memory space
- Facts too are often numeric properties and can usually be represented as integers (contrast to dimensional attributes which are usually long text strings)
- This *space efficiency* is critical to the memory space consumption of the data warehouse

Keys

- Choice the data warehouse keys to be meaningless surrogate keys
  - Let a surrogate key be a simple integer
  - 4-byte (--------,--------,--------,--------) can contain 2^{32} values (> 2 billion positive integers, starting with 1)

Keys

- Use surrogate keys also for the Time dimension
  - SQL-based date key, is typically 8 bytes, so 4 bytes are wasted
  - bypassing joins leads to embedding knowledge of the calendar in the application, rather than reading it from the time dimension
  - it is not possible to encode a data stamp as “I do not know”, “It has not happen yet”, etc
- Avoid smart keys
- Avoid production keys
  - production may decide to reuse keys
  - the company may acquire a competitor and thereby change the key building rules
  - changed record, but deliberately not changed key

Slowly Changing Dimensions

For example, the product or customer dimension
The assumption: the key does not change, but some of the attributes does.

- Type 1: Overwrite the dimension record 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

Overwrite the old value of an attribute with a new one

e.g.

12334	Mary	Jones	single	married

+ easy to implement

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

Type 2

Create a new additional dimension record

- A generalised (surrogate) key is required (which is a responsibility of the data warehouse team)

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

Rapidly Changing Dimensions

From the previous slides: What is slow?

What if the changes are fast?

Must a different design technique be used?

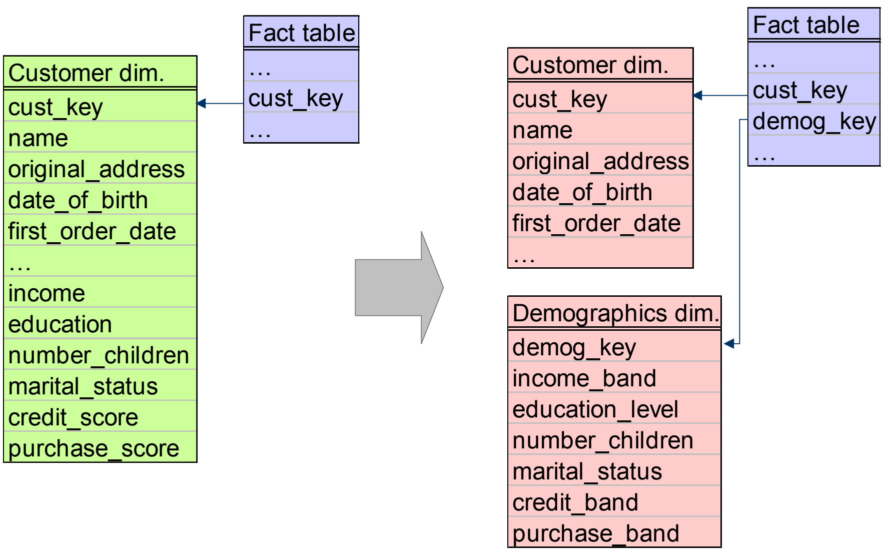
- Small dimensions:
  - the same technologies as for slowly changing dimensions may be applied
- Large dimensions:
  - the choice of indexing techniques and data design approaches are important
  - find suppress duplicate entries in the dimension
  - do not create additional records to handle the slowly changing dimension problem

Rapidly changing very large dimensions

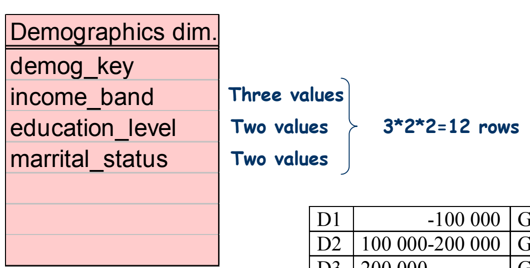
- Break off some of the attributes into their own separate dimension(s), a demographic dimension(s).
  - force the attributes selected to the demographic dimension to have relatively small number of discrete values
  - build upp the demographic dimension with all possible discrete attributes combinations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

NB! The demographic attributes are the one of the heavily used attributes. Their values are often compared in order to identify interesting subsets.

Demographic Minidimension

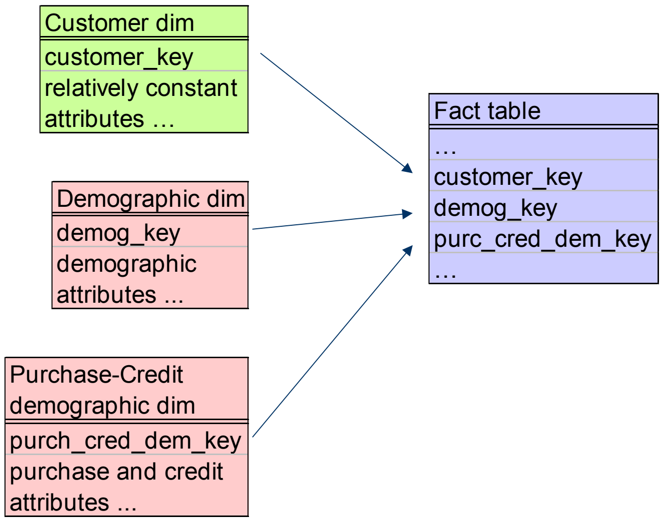


Demographic Minidimension



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*

Two Demographic Minidimensions



Demographic 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 (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
  - slower down the browsing
- What if the fact table (connecting the demographic minidimension with the customer dimension) is sparse?

Demographic Minidimension

- What to do if the fact table (connecting the demographic minidimension with the customer dimension) is sparse?

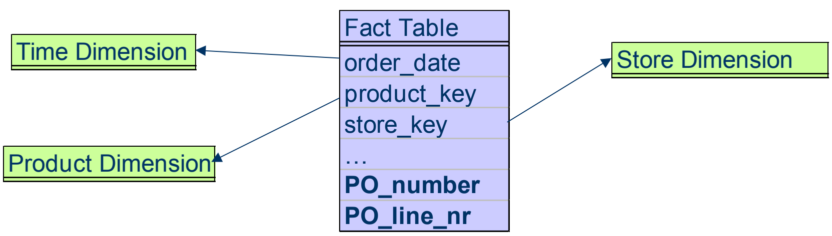
  - Define a demographic transaction event, i.e., introduce a new fact table

  or

  - Add a current demographic key to the customer dimension table

Degenerate Dimension

- A degenerate dimension is represented by a dimension key attribute(s) with no corresponding dimension table
- Occurs usually in line-item oriented fact table design



Junk Dimensions

When a number of miscellaneous flags and text attributes exist, the following design alternatives should be avoided:

- Leaving the flags and attributes unchanged in the fact table record
- Making each flag and attribute into its own separate dimension
- Stripping out all of these flags and attributes from the design

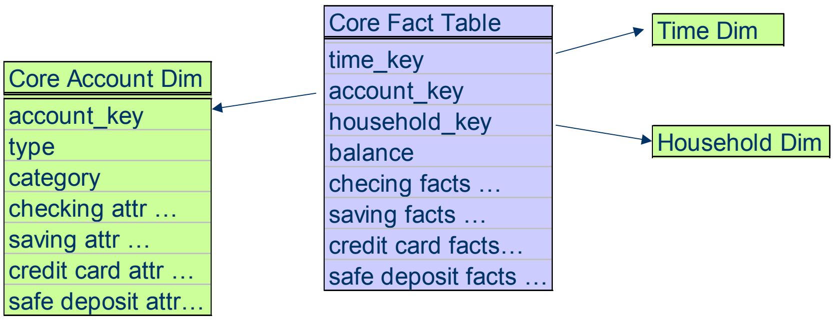
A better alternative is to create a junk dimension.
A junk dimension is a convenient grouping of flags and attributes to get them out of a fact table into a useful dimensional framework

Heterogeneous Products

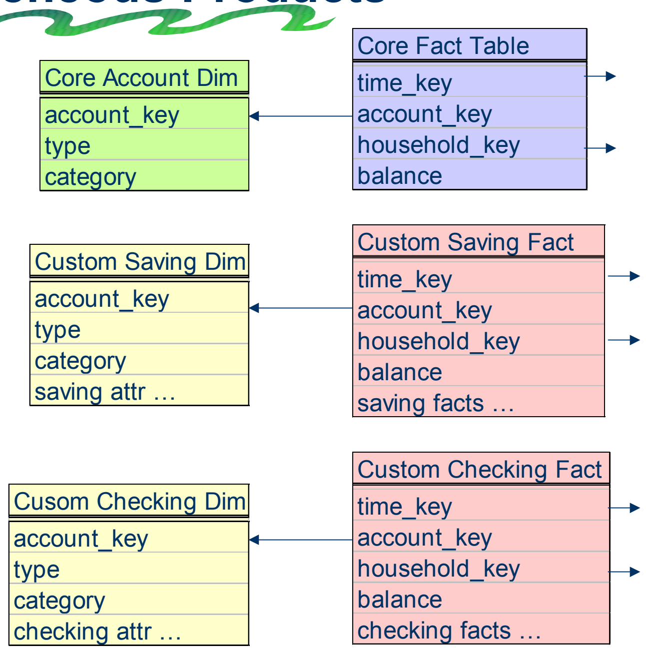
Some products have many, many distinguishing attributes and many possible permutations (usually on the basis of some customised offer). This results in immense product dimensions and bad browsing performance

- 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 core facts on the products types are kept in a *core fact table* (but can also be copied to the conformed fact tables)

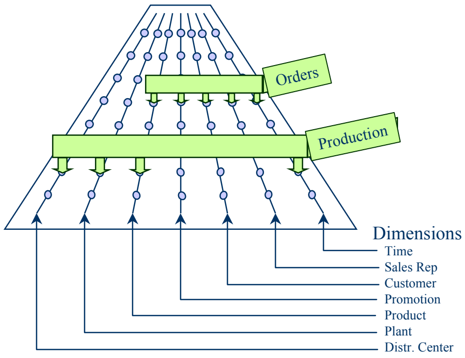
Heterogeneous Products



Heterogeneous Products



The Data Warehouse Bus



Dimensional modelling vs. ER-modelling

Entity-relationship modelling

- a logical design technique to eliminate data redundancy to keep consistency and storage efficiency
- makes transaction simple and deterministic
- ER models for enterprise are usually complex, e.g. they often have hundreds, or even thousands, of entities/tables

Dimensional modelling

- a logical design technique that present data in a intuitive way and that allow high-performance access
- aims at model decision support data
- easier to navigate for the user and high performance

Why dimensional modelling?

- the logical model is easy understand
- a predictable standard framework for end user applications
- the logical design can be done nearly independent of expected query pattern
- handle changes easy - at least adding new dimensional attributes
- high performance "browsing" across the attributes, eliminating joins and make use bit vector indexes
- strategy to handling aggregates, e.g. summery records that are logical redundant with base table to enhance query performance
- the database engine can make strong assumption how to optimise
- strategies for handling slowly changing dimensions, heterogenous products, event-handling ("factless fact tables")

To be continued