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

Keywords
degenerate dimension, junk dimension, heterogeneous products, many-to-many relationships, factless fact table, bridge table, family of stars, value and circle chains, data warehouse bus, the design process, aggregates
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

Core Fact Table
- time_key
- account_key
- household_key
- balance
- checking facts …
- saving facts …
- credit card facts …
- safe deposit facts …

Time Dim

Household Dim

Core Account Dim
- account_key
- type
- category
- checking attr …
- saving attr …
- credit card attr …
- safe deposit attr …

Custom Saving Dim
- account_key
- type
- category
- saving attr …

Custom Checking Dim
- account_key
- type
- category
- checking attr …

Core Fact Table
- time_key
- account_key
- household_key
- balance

Custom Saving Fact
- time_key
- account_key
- household_key
- balance
- saving facts …

Custom Checking Fact
- time_key
- account_key
- household_key
- balance
- checking facts …
Dealing with many-to-many relationships

- Many to many relationships (M-to-M) between entities (tables) are difficult to deal with in any database design situation. E.g. A customer can have many accounts and an account may belong to many customers.

- A new table can be created to capture the relationship between the tables.

- Many to many relations between dimensional tables in a star-join schema can be handle by creating a factless fact table or a bridge table.

Factless fact tables

- Some fact tables quite simply have no measured facts!
- Are useful to describe events and coverage, i.e. the tables contain information that something has/has not happened.
- Often used to represent many-to-many relationships.
- The only thing they contain is a concatenated key, they do still however represent a focal event which is identified by the combination of conditions referenced in the dimension tables.
- There are two main types of factless fact tables:
  - event tracking tables
  - coverage tables
Factless fact tables

Event tracking tables
- records events, e.g. records every time a student attends a course, or people involved in accidents and vehicles involved in accidents

Coverage tables
- description of something that did not happen, e.g. which product did not sell during a promotion campaign.

Bridge table
Problem: There are multiple diagnosis involved in the billed amount? The users of the dw want to know for how much money a certain diagnosis is billed for?
Bridge table

- Handle open ended many-value attribute, i.e. dimensions with many values which are not knowable before the fact table is created
- A weighting factor in the bridge table to add up the billed amount correctly for each diagnosis

```
<table>
<thead>
<tr>
<th>Bridge table</th>
<th>Diagnosis Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagnosis_group_key</td>
<td>diagnosis_key</td>
</tr>
<tr>
<td>weighting_factor</td>
<td>other_attributes ...</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Billable Patient Fact</th>
<th>Patient Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_key</td>
<td>patient_key</td>
</tr>
<tr>
<td>diagnosis_group_key</td>
<td>billed_amount</td>
</tr>
</tbody>
</table>
```

Bridge tables

```
SU

Faculty of Social Sc.

FEK

Faculty of Low

DSV

Dept. of Low

K2lab

Syslab

Peter

Sten

Petia

Erik
```
Bridge tables

- How many books are sold to SU and their employees?
- How many books are sold to DSV and their employees?

**SELECT C.Name, sum(Ex)
FROM Customer AS C, Bridge AS B,
Sale AS S
WHERE S.Cust = B.Child
AND B.Parent = C.Name
AND Level >= 0
GROUP BY C.Name**
### Bridge tables

<table>
<thead>
<tr>
<th>Date</th>
<th>Cust</th>
<th>Book</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-Feb</td>
<td>Petia</td>
<td>Kimball</td>
<td>1</td>
</tr>
<tr>
<td>3-Feb</td>
<td>Erik</td>
<td>Kimball</td>
<td>1</td>
</tr>
<tr>
<td>4-Feb</td>
<td>Peter</td>
<td>Elmasri</td>
<td>2</td>
</tr>
<tr>
<td>15-Mar</td>
<td>DSV</td>
<td>Kimball</td>
<td>80</td>
</tr>
<tr>
<td>30-Mar</td>
<td>F of So Sc</td>
<td>Booth</td>
<td>40</td>
</tr>
<tr>
<td>3-Apr</td>
<td>Petia</td>
<td>Cood</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parent</th>
<th>Child</th>
<th>Level</th>
<th>Leaf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Petia</td>
<td>Petia</td>
<td>0</td>
<td>y</td>
</tr>
<tr>
<td>Syslab</td>
<td>Petia</td>
<td>1</td>
<td>y</td>
</tr>
<tr>
<td>DSV</td>
<td>Petia</td>
<td>2</td>
<td>y</td>
</tr>
<tr>
<td>F of So Sc</td>
<td>Petia</td>
<td>3</td>
<td>y</td>
</tr>
<tr>
<td>SU</td>
<td>Petia</td>
<td>4</td>
<td>y</td>
</tr>
<tr>
<td>Syslab</td>
<td>Syslab</td>
<td>0</td>
<td>n</td>
</tr>
<tr>
<td>DSV</td>
<td>Syslab</td>
<td>1</td>
<td>n</td>
</tr>
<tr>
<td>F of So Sc</td>
<td>Syslab</td>
<td>2</td>
<td>n</td>
</tr>
<tr>
<td>SU</td>
<td>Syslab</td>
<td>3</td>
<td>n</td>
</tr>
<tr>
<td>DSV</td>
<td>DSV</td>
<td>0</td>
<td>n</td>
</tr>
<tr>
<td>F of So Sc</td>
<td>DSV</td>
<td>1</td>
<td>n</td>
</tr>
<tr>
<td>SU</td>
<td>DSV</td>
<td>2</td>
<td>n</td>
</tr>
<tr>
<td>F of So Sc</td>
<td>F of So Sc</td>
<td>0</td>
<td>n</td>
</tr>
<tr>
<td>SU</td>
<td>F of So Sc</td>
<td>1</td>
<td>n</td>
</tr>
<tr>
<td>SU</td>
<td>SU</td>
<td>0</td>
<td>n</td>
</tr>
</tbody>
</table>

### 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 facilitate drill-across.

- A **conformed dimension** is a dimension that means the same thing with every possible fact table to which it is joined.

- **Conformed dimensions** are either identical or strict mathematical subsets of the most granular detailed dimension.
Value chains as families of star-join schemas

- There are two sides to the value chain
  - the demand side - the steps needed to satisfy the customers’ demand for the product
  - the supply side - the steps needed to manufacture the products from original ingredients or parts
- The chain consists of a sequence of inventory and flow star-join schemata
- joining the different star-join schemata is only possible when two sequential schemata have a common, identical dimension
- Sometimes the represented chain can be extended beyond the bounds of the business itself

Supply Chain
- Row material production
- Ingredient purchasing
- Ingredient delivery
- Ingredient inventory
- Bill of materials
- Manufacturing process control
- Manufacturing costs
- Packaging
- Trans-shipping to warehouse
- Finished goods inventory

Demand Chain
- Finished goods inventory
- Manufacturing shipments
- Distributor inventory
- Distributor shipments
- Retail inventory
- Retail sales
Value circles

- Clinics
- Long Term Care Facilities
- Physician Offices
- Hospitals
- Pharmacies
- Insurance Companies
- Employers
- Laboratories

A family of stars

- Star schema 1
- Star schema 2
- Star schema 3
- Star schema 4

Common dimension
Common dimension
Common dimension
Common dimension
Common dimension
1 Choose a business process to model

A business process is a major operational process in an organisation, that is supported by some kind of a legacy system(s) from which data can be collected, e.g., orders, invoices, shipments, inventory.

2 Choose the grain of the business process

The grains is the level of detail at which the data is represented in the DW. Typical grains are individual transactions, individual daily (monthly) snapshots.

3 Choose the dimensions that will apply to each fact table record

Typical dimensions are time, product, customer, store, etc.

4 Choose the measured facts that will populate fact table

E.g., quantity sold, dollars sold
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

Consider the following questions

• How much total business did my newly remodelled stores do compared with the chain average?
• How did leather goods items costing less than $5 do with my most frequent shoppers?
• What was the ratio of non-holiday weekend days total revenue to holiday weekend days?
Aggregation

- 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

Case

NEW CLOTHES is a retailer of clothes. It has a total of 85 stores in the Nordic countries. The management of NEW CLOTHES wants the ability to analyse the sales of different products. It should be possible to analyse which products that have been sold in which stores for a certain day, week, month and year. There is no need to analyse every transaction in the stores, it is only necessary to analyse the "amount" in SEK och "quantity sold" for each product, per store, per day. The products have been grouped in different categories, e.g. trousers and socks, and the management should be able to analyse the sales of different product categories. It is also important to analyse the sales of different brands, like Levi's or Wrangler.

NEW CLOTHES’ stores are spread all over the Nordic countries. Every country is divided into regions and the management will be able to analyse the sales of stores in a certain region and the sales of stores in a certain country. It should also be possible to analyse data about which city the stores are located in and who is responsible for each store. However, there are no data about the customers available. Therefore, the customers are not part of the analysis of the sales.

Furthermore, the management of NEW CLOTHES wants to be able to analyse the product returns made by customers. It should be possible to analyse each return in detail, like “returned quantity” for a specific product and “returned amount” in SEK. It should also be possible to analyse returns (quantity and amount) per brand and per store.

A customer has the right to return (öppet köp) a product within a week after a purchase. He/she can then choose whether he/she wants to receive the money back, or to exchange the returned product to another one. The customer has also the right to return a product even longer time after the purchase, if it turns out that the product in question has some defect. Even in this case the money can be returned or a new product selected. It is of interest to the management to know which of those options a customer chooses for a return he/she makes. The management also wants to analyse the type of return, e.g., whether it was a defect in the product or whether the customer simply changed his/her mind, and the reason for the return (öppet köp), e.g. wrong size, or wrong colour, and in case of defect, e.g., zip broken, discoloration resulting from washing. When the customer return a product he/she must supply the name and address (postadress). The management wants to use the names to analyse if a certain customer makes several returns and the addresses to see how often customers from a certain housing area make returns.