Using Design Guidelines to Improve Data Warehouse Logical Design

Verónika Peralta - Raúl Ruggia

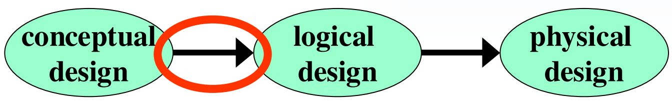
Universidad de la República

URUGUAY

vperalta@fing.edu.uy - ruggia@fing.edu.uy

Context

- **DW design process:**



Generate the DW relational schema from a conceptual schema

Automate the generation process

- **Two main aspects:**
  - Mapping the conceptual structures to the logical ones.
    - Define the relational structures.
  - Considering implementation-oriented requirements.
    - Performance, storage constraints, workload.

The problem

- **Improve DW logical schema taking into account implementation requirements:**
  - Support workload (performance, etc).
    - Not cover all the aspects to take into account.
  - Implementation-oriented requirements.
    - Very hard to formalize.

➔ Define some kind of specification:
- Represent implementation-oriented requirements.
- Enable automated processing.

Existing approaches

- **Generation from conceptual schemas:**
  - [Golfarelli-Rizzi], [Cabibbo-Torlone], [Hahn et al]...
  - Focus on mappings: conceptual → logical.
  - Do not cover implementation-related requirements.
  - Do not provide flexible enough generation techniques
    - Required to take into account such requirements.
- **[Golfarelli-Rizzi]:**
  - Also proposes vertical fragmentation based on query workload.

Goals of this work

- **Formalize implementation-oriented information:**
  - to be used in an automated design process.
- **This involves:**
  - Identifying the most relevant information.
    - Expressiveness vs. treatability vs. ease of use.
  - Specifying such information.
    - Formalism.
    - Graphical notation.
  - Using it to generate the logical schema.

Outline

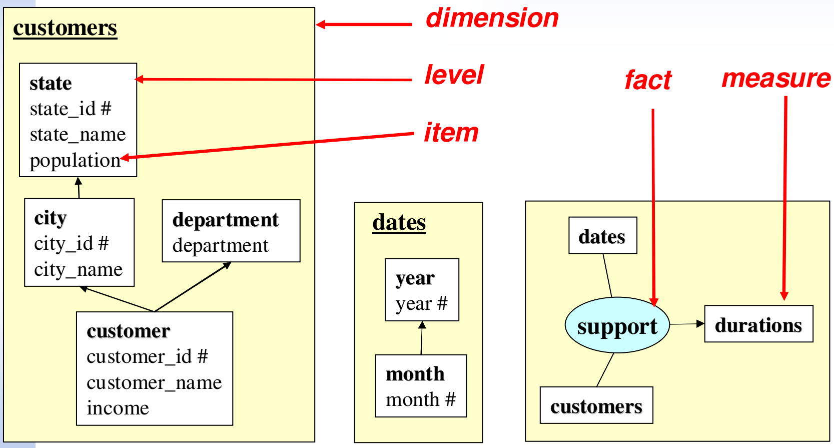
- **Design guidelines.**
  - Examples.
  - The formalism.
  - Using the guidelines.
- **The DW design environment.**
- **Towards automation.**
- **Conclusion**

Introducing design guidelines

- **Guidelines are assertions**
  - Represent design strategies to solve implementation-related requirements.
- **Properties:**
  - Ease of use. Easily defined by the designer.
  - Treatable for automation. Interpreted by a process.
  - Expressive. Allow solving non-trivial cases.
- **Three types of guidelines:**
  - Vertical fragmentation of dimensions.
  - Aggregate materialization.
  - Horizontal fragmentation of facts.

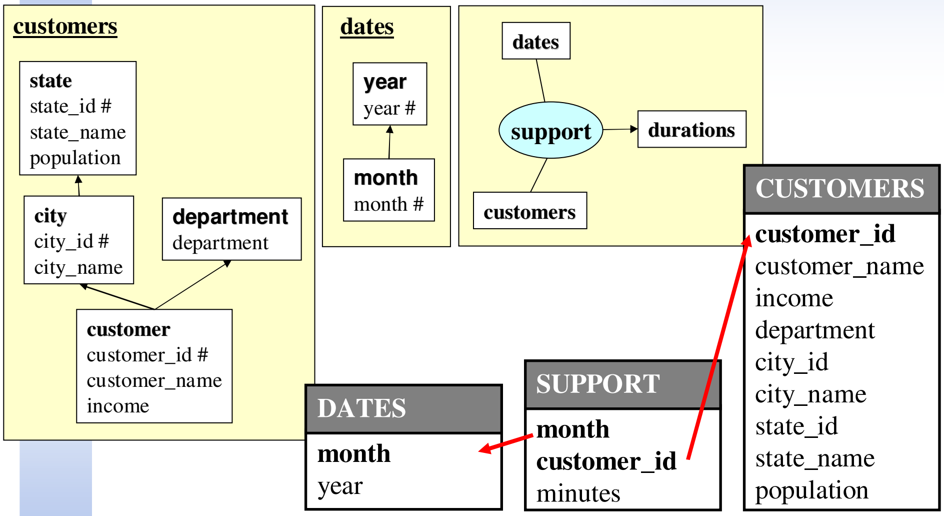
Example

Conceptual schema (CMDM notation)



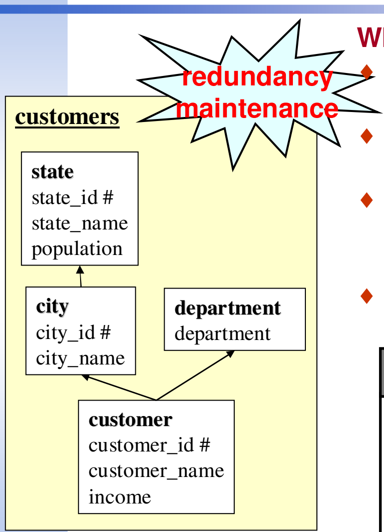
Example

Star schema



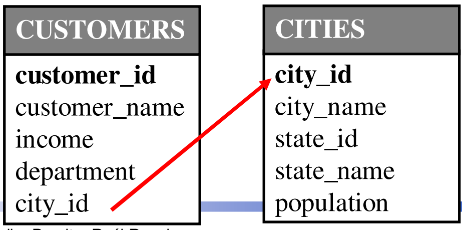
Example

Design situations Normalizing



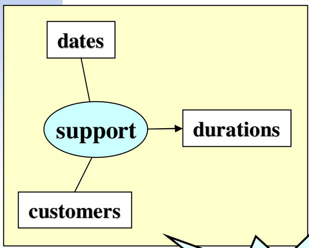
When and why:

- **Complex or big dimensions**
  - 100.000 customers, 100 cities.
- **Versioning**
  - Trace the history of the cities.
- **Different query profiles**
  - Geographical distribution of calls.
  - Supervise customers by departments.
- **Different access frequency**
  - Most accessed: department.



Example

Design situations
Aggregating and fragmenting



SUPPORT
month
customer_id
minutes

When and why:

- **High-summarized data**
  - Total minutes for each year and city.
- **Different query profiles**
  - Frequent queries access data of 2003.
- **Complex queries**
  - Quantity of customers that make long calls, classifying durations in ranges.
- **Additivity problems**
  - Quantity of customers for each year.

CUSTOMER_Q
month
city_id
duration_range
customer_quantity

CUSTOMER_YQ
year
city_id
duration_range
customer_quantity

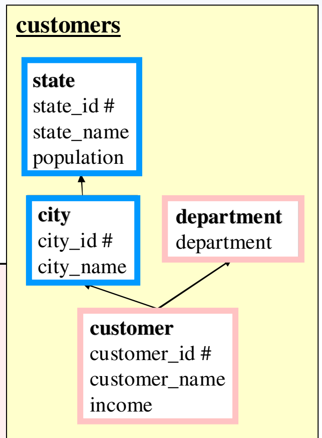
Guidelines

Vertical Fragmentation of Dimensions

- **The designer indicates:**
  - The set of levels to store together. Fragments

fragment 1: customer and department.

fragment 2: city and state.



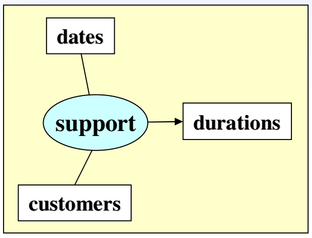
$$
\begin{aligned}
&\text{SchFragments} \subseteq \{ <\text{Fname}, D, Ls> / \\
&\quad \text{Fname} \in \text{Strings} \wedge \\
&\quad D \in \text{SchDimensions} \wedge \\
&\quad Ls \subseteq \text{GetLevels}(D) \wedge \\
&\quad \forall A,B \in Ls \,.\, (\\
&\qquad <A,B> \in \text{GetHierarchies}(D) \vee \\
&\qquad <B,A> \in \text{GetHierarchies}(D) \vee \\
&\qquad \exists C \in Ls \,.\, (<A,C> \in \text{GetHierarchies}(D) \wedge <B,C> \in \text{GetHierarchies}(D)))
\end{aligned}
$$

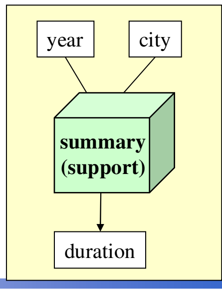
Guidelines

Aggregate Materialization

- **The designer indicates:**
  - The derived fact tables to materialize. Cubes

    level of detail: year, city, duration
    measure: duration





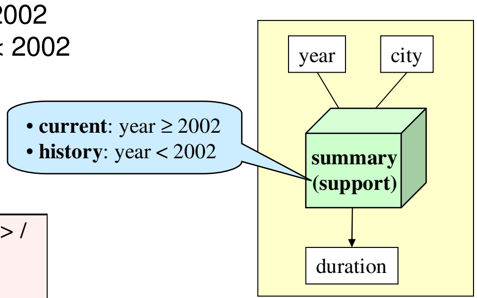
SchCubes $\subseteq$ { <Cname, R, Ls, M> /
Cname $\in$ Strings $\wedge$
R $\in$ SchFacts $\wedge$
Ls $\subseteq$ { L $\in$ GetLevels(D) / D $\in$ GetDimensions(R) } $\wedge$
M $\in$ (Ls $\cup$ $\perp$)
}[1]

Guidelines

Horizontal Fragmentation of Facts

- **The designer indicates:**
  - The conditions to fragment fact data. Strips

    current data: year ≥ 2002
    historical data: year < 2002



SchStrips $\subseteq$ { <Sname, C, Pred> /
Sname $\in$ Strings $\wedge$
C $\in$ SchCubes $\wedge$
Pred $\in$ Predicates(GetItems(C))
}[1]

Definition of guidelines

- **By means of the guidelines, the designer defines:**
  - A set of fragments for each dimension.
    - Performance vs. redundancy (maintenance).
    - Workload: data queried together, access frequencies.

  ✓ **Complex or big dimensions**
  ✓ **Versioning**
  ✓ **Different query profiles**
  ✓ **Different access frequency**

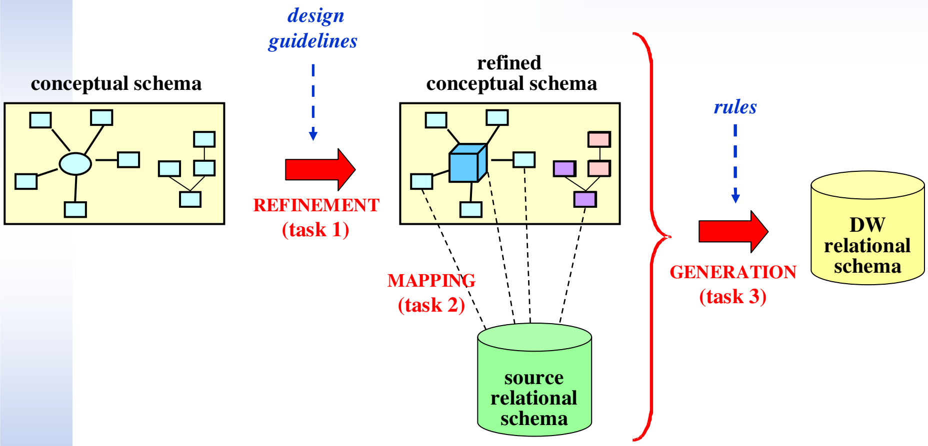
  - A set of cubes for each fact.
    - Storage vs. performance constraints.
    - Additivity.

  ✓ **High-summarized data**
  ✓ **Complex queries**
  ✓ **Additivity problems**

  - A set of strips for each cube.
    - Workload.

  ✓ **Different query profiles**

DW logical design environment

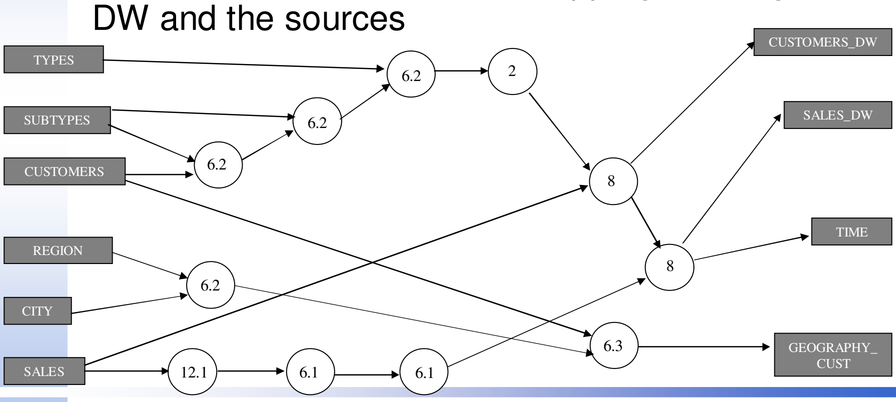


Automating logical design

- **Objective:**
  - Express DW schema as a set of transformations on source schemas
  - Automate the secuence of operations that may do a designer
    - Based on his experience
    - Following design tips
- **Example:**
  - When we want to denormalize dimensions and target data is distributed in several tables then we have to join those tables
- **Idea:**
  - Having schema transformations and design rules

Automating logical design

- **Schema transformations**
  - Refer to common transf. in DW design
  - A transf. trace indicates the mappings among the DW and the sources

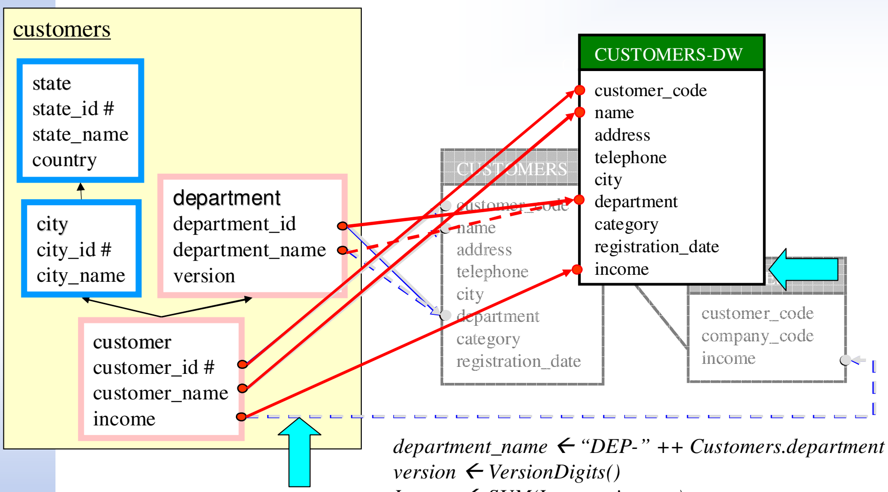


Automating logical design

- **Input:**
  - Conceptual schema (express user requirements)
  - Design guidelines (express non functional requirements)
  - Mappings to source databases (indicate where to extract data)
- **Process:**
  - Choose the schema transformations to apply
  - Apply the schema transformations
- **Output (result of applying transformations):**
  - DW relational schema
  - Design trace

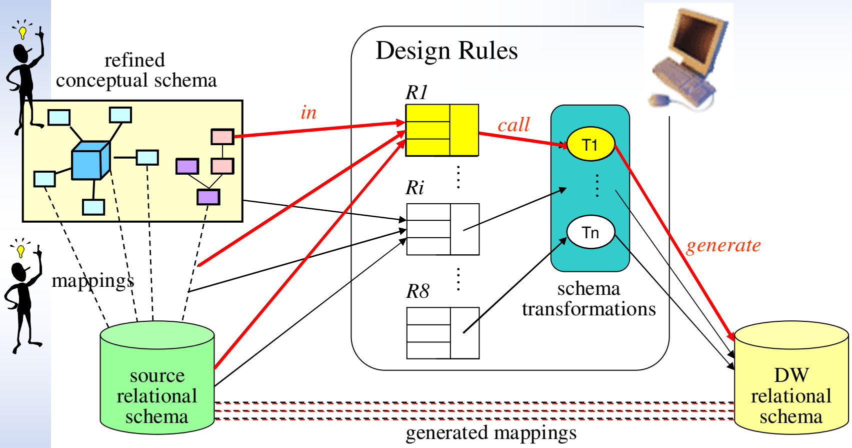
Automating logical design

Rule
Aggregate Calculate



department_name ← *"DEP-" ++ Customers.department*
version ← *VersionDigits()*
~~*Income* ← *SUM(Incomes.income)*~~

Automating logical design



Conclusion

- **A formalism for specifying design guidelines:**
  - Take into account implementation-related requirements.
  - Three type of guidelines:
    - Aggregate materialization.
    - Vertical fragmentation of dimensions.
    - Horizontal fragmentation of facts.
  - A trade-off between expressiveness, ease of use and treatable for automation.
  - The set can be extended.
  - A flexible way to express design strategies.
- **A prototype of a CASE tool for DW logical design.**