Advanced Data Management Technologies

Unit 7 — Changing Dimensions

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE

Acknowledgements: I am indebted to M. Böhlen for providing me the lecture notes.

Outline

1. Slowly Changing Dimensions

2. Rapidly Changing Dimensions

Outline

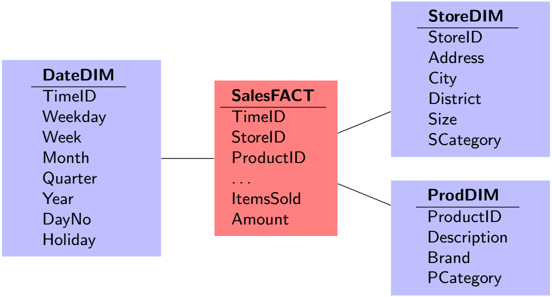
1. Slowly Changing Dimensions

2. Rapidly Changing Dimensions

Changing Dimensions

- So far, we have implicitly assumed that dimensions are stable over time.
    - At most, new rows in dimension tables are inserted.
    - The existing rows do not change.
- This assumption is not valid in practice.
    - The phenomenon is called slowly changing dimensions.
    - The intuition is that dimension information changes, but changes are (relatively) rare.
- We will look at a number of techniques for handling changes in dimensions.
- Schema changes are not considered.
    - Then it becomes really funny!

Changing Dimensions/2



- Descriptions of stores and products vary over time.
  - A store is enlarged and changes Size.
  - A product changes Description.
  - Districts are changed.
- Problems
  - If we update the dimensions, wrong information will result.
  - If we don't update the dimensions, the DW is not up-to-date.

Solution 1: Overwrite Old Values/1

- **Solution 1:** Overwrite the old values that change in the dimension tables.
  - Today-for-yesterday viewpoint (up-to-date): all the events including past ones are always interpreted from the viewpoint of the current dimension values, without tracking previous instances.
  - Old facts point to rows in the dimension tables with incorrect information.
  - New facts (i.e., inserted after the dimension rows changed) point to rows with correct information.
- Pros
  - Easy to implement
  - Ideal if the changes are due to erroneous registrations.
  - In some cases, the imprecision can be disregarded.
- Cons
  - The solution does not solve the problem of capturing change.

Solutions 1: Overwrite Old Values/2

SalesFACT

StoreID	...	ItemsSold	...
001		2000	

StoreDIM

StoreID	...	Size	City
001		250	Chikago

⇓

SalesFACT

StoreID	...	ItemsSold	...
001		2000	

StoreDIM

StoreID	...	Size	City
001		450	Chicago

⇓

SalesFACT

StoreID	...	ItemsSold	...
001		2000	
001		2500	

StoreDIM

StoreID	...	Size	City
001		450	Chicago

Solution 2: Versioning of Rows/1

- **Solution 2:** Versioning of rows with changing attributes.
  - An event stored into a fact table is associated with the version of the dimesion row that was valid when that event took place.
  - Today-or-yesterday viewpoint (historical truth): Each event is analyzed according to the dimension values at the time when the event occurred.
  - Yields larger dimension tables
- Pros
  - Correct information captured in DW
  - No problems when formulating queries
- Cons
  - It is not possible to capture the development over time of the subjects the dimensions describe since time of change is not recorded.

Solution 2: Versioning of Rows/2

SalesFACT

StoreID	...	ItemsSold	...
001		2000	

StoreDIM

StoreID	...	Size	...
001		250	

⇓

SalesFACT

StoreID	...	ItemsSold	...
001		2000	

StoreDIM

StoreID	...	Size	...
001		250	
002		450	

⇓

SalesFACT

StoreID	...	ItemsSold	...
001		2000	
002		2500	

StoreDIM

StoreID	...	Size	...
001		250	
002		450	

Solution 3: Timestamping of Rows/1

- **Solution 3:** Versioning of rows with changing attributes like in Solution 2 + timestamping of rows to support every temporal analysis scenario.
- Additionally yesterday-for-today viewpoint (rollback) is supported: after the user has set a specific date, the dimension table rows that were valid at that time are found.
- Add a new tuple to the dimension table whenever an attribute changes.
- Required features:
  - A couple of timestamps specifying the validity interval of tuples
  - A master attribute that specifies the key value of the tuple from which each tuple stems
- Pros
  - Correct information captured in DW as time is stored
- Cons
  - Larger dimension tables

Solution 3: Timestamping of Rows/2

SalesFACT

StoreID	TimeID	...	ItemsSold	...
001	234		2000	

StoreDIM

StoreID	Size	From	To	Master
001	250	98	-	1
002	400	98	-	2

⇓

SalesFACT

StoreID	TimeID	...	ItemsSold	...
001	234		2000	

StoreDIM

StoreID	Size	From	To	Master
001	250	98	99	1
002	400	98	-	2
003	450	00	-	1

⇓

SalesFACT

StoreID	TimeID	...	ItemsSold	...
001	234		2000	
002	456		2500	

StoreDIM

StoreID	Size	From	To	Master
001	250	98	99	1
002	400	98	-	2
003	450	00	-	1

Solution 3: Timestamping of Rows/3

- Product descriptions are versioned, when products are changed, e.g., new package sizes.
- New facts can refer to both the newest and older versions of products, as old versions are still in the stores.
- Thus, the Time value for a fact should not necessarily be between the From and To values in the Product dimension row.
- This is unlike changes in Size for a store, where all facts from a certain point in time will refer to the newest Size value.
- This is also unlike alternative categorizations (e.g, department) that one wants to choose between.

Solution 3b: Special Facts/1

- **Solution 3b:** Solution 2 + special facts for capturing changes in dimensions via the Time dimension.
  - When a change occurs and there is no simultaneous, new fact referring to the new dimension row, a new special fact is created that points to the new dimension row and thus timestamps the row via the fact row's reference to the Time dimensions.
- Pros
  - It is possible to capture the development over time of the subjects that the dimensions describe
  - Smaller dimension tables than in solution 3a
- Cons
  - Complexity of working with this solution

Solution 3b: Special Facts/2

SalesFACT

StoreID	TimeID	. . .	ItemsSold	. . .
001	234		2000	

StoreDIM

StoreID	. . .	Size	. . .
001		250	

⇓

SalesFACT

StoreID	TimeID	. . .	ItemsSold	. . .
001	234		2000	
002	345		-	

StoreDIM

StoreID	. . .	Size	. . .
001		250	
002		450	

⇓

SalesFACT

StoreID	TimeID	. . .	ItemsSold	. . .
001	234		2000	
002	345		-	
002	456		250	

StoreDIM

StoreID	. . .	Size	. . .
001		250	
002		450	

Outline

1. Slowly Changing Dimensions

2. Rapidly Changing Dimensions

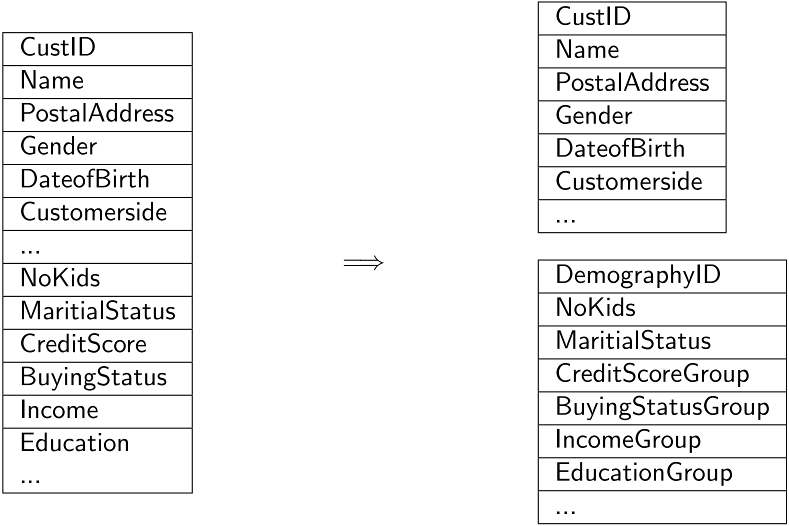
Rapidly Changing Dimensions/1

- Difference between slowly and rapidly is subjective.
- Solution 2 is often still feasible
- A problem might be the size of the dimension
  - e.g., an Employee dimension with 100,000 employees, each using 2K and many changes every year.
- Other typical examples of (large) dimensions with many changes are Product and Customer.
  - Some Customer dimensions can have 10M customers.
  - Use Solution 2 and suitable indexing!
- Monster dimensions: The more attributes in a dimension table, the more changes per row can be expected ⇒ dimension might become very large.
  - e.g., a Customer dimension with 100M customers and many attributes.

Rapidly Changing Dimensions/2

- Solution with Mini-dimension
  - Make a mini-dimension with the often-changing attributes, e.g., demographic attributes.
  - Convert (numeric) attributes with many possible values into attributes with few possible values, representing groups of the original values.
  - Insert rows for all combinations of values from these new domains.
    - With 6 attributes with 10 possible values each, the dimension gets 1,000,000 rows.
    - Alternatively, (combination) rows can be inserted when needed.
  - If the mini-dimension is too large, it can be split into two or more mini-dimensions.

Rapidly Changing Dimensions/3



CustID
Name
PostalAddress
Gender
DateofBirth
Customerside
...
NoKids
MaritialStatus
CreditScore
BuyingStatus
Income
Education
...

⟹

CustID
Name
PostalAddress
Gender
DateofBirth
Customerside
...

DemographyID
NoKids
MaritialStatus
CreditScoreGroup
BuyingStatusGroup
IncomeGroup
EducationGroup
...

Rapidly Changing Dimensions/5

- Pros
  - DW size (dimension tables) is kept down.
  - Changes in a customer's demographic values do not result in changes in the customer dimension.
    - Rows must be inserted only into the mini-dimension.
- Cons
  - More dimensions and more keys in the star schema.
  - Using value groups gives less detail.
  - The construction of groups is irreversible and makes it hard to make other groupings.
  - Navigation of customer attributes is more cumbersome as these are in more than one dimension.
    - An ActualDemography attribute can be added to the dimension with the stable values.

Summary

- Multidimensional models realized as star schemas support change over time to a large extent.
- This is important!
  - Applications change.
  - The modeled reality changes.
- Three techniques for handling changing dimensions.
  - Overwrite old values: simplest, but captures only up-to-date view
  - Versioning of rows: very often a good tradeoff, captures historical truth
  - Timestamping of rows: most advanced solution, support rollback.
- Rapidly changing dimensions and monster dimensions might occur, which might lead to very large dimensions
- Mini-dimensions is a solution to solve the size problem.