BI Data Modeling: MultiProviders and InfoSets

Version 1.0
July 4, 2006
Table of Contents:

1 Do’s and Don’ts for MultiProviders and InfoSets

2 MultiProvider Scenarios versus Partitioning Characteristics

3 Data Models Using InfoSets
   3.1 Second-Order Navigational Attributes
   3.2 Data Model Examples for Temporal Joins

4 Performance Aspects of MultiProviders and InfoSets

5 List of Documents Related to MultiProviders and InfoSets
MultiProviders and InfoSets

An InfoProvider is a BI Content object for which BI queries can be created or executed in the BEx. InfoProviders are the objects or views that are relevant for reporting. A MultiProvider as well as an InfoSet do not physically store data, but display logical views.

- A MultiProvider builds up a data union of basic InfoProviders. The complete data of all basic InfoProviders are available for reporting. A MultiProvider is interpreted at runtime as independent BI queries on each basic InfoProvider where the results are merged into a single result set.
- An InfoSet builds up a data join of basic InfoProviders. The valid combination of records from the basic InfoProviders is determined by the join condition of the InfoSet.

1 Do’s and Don’ts for MultiProviders and InfoSets

MultiProviders

- When the reporting scenario is to be extended, use a MultiProvider as central interface between query definition and basic InfoProviders. When another InfoProvider is added to the MultiProvider definition, the technical name of a query based on the MultiProvider remains unchanged.
- Use a MultiProvider to reduce the size of the basic InfoProviders. Advantages: parallel access to underlying basic InfoProviders, load balancing, resource utilization, query pruning.
- Make sure that your MultiProvider only retrieves data from relevant basic InfoProviders at query runtime by
  - Using constants in the design of the basic InfoProviders
  - Using different key figures in the design of the basic InfoProviders
  - Using characteristic 0INFOPROV when designing a query on the MultiProvider
- Are you planning to use a MultiProvider? If so, you have to ensure that the characteristics you want to report exist in all basic InfoProviders.
- Do not use more than one non-cumulative InfoCube (InfoCube with at least one non-cumulative key figure) because this could lead to incorrect query results.
- Do not use calculations before aggregation on MultiProvider because this may lead to wrong query results.
- Do not combine basic InfoProviders having inhomogeneous data models in a MultiProvider. Use the report-report interface between queries defined on the basic InfoProvider instead.
- Avoid using only parts of compound characteristics in the constituent basic InfoProvider of a MultiProvider. For more information, see SAP note 702542.

InfoSets

- Do not use more than 10 InfoProviders in one InfoSet. It is better to create multiple InfoSets depending on reporting needs.
- Do not use more than 10 joins in one InfoSet (especially if you expect high a data volume).
- InfoSet queries can be used for DataStore objects without the activated BEx Reporting indicator. See also the Performance Aspects section of this document.
- Do not use calculations before aggregation on InfoSet because this may lead to wrong query results.
- If there are InfoSets with time-dependent master data, do not restrict the data by the fields Valid from (0DATEFROM) and Valid to (0DATETO). See also an example in the Data Models using InfoSets section of this document.
2 MultiProvider Scenarios versus Partitioning Characteristics

During the BI data modeling phase, there are many key figures such as:

- Plan / Actual / Target Costs,
- Planned Sales Amount, Actual Sales Amount,
- Statistical / Actual Amount,
- Total Sales Amount in Local Currency, Total Sales Amount in Document Currency.

In the following section, three different scenarios are described, which can be used as a BI data model for such key figures. In conclusion, we valuate the three scenarios in a comparing matrix with respect to their possible use cases.

Scenario 1: InfoCube Contains All Key Figures (key figure data model)

All key figures of the data model are defined as InfoObject and are added to the fact table of one InfoCube.

Examples

- The 3 key figures Plan Costs of Version 1, Plan Costs of Version 2, and Actual Costs are included in one InfoCube.
- The 2 key figures Total Sales Amount in Local Currency and Total Sales Amount in Document Currency are included in one InfoCube.
- Several detailed views corresponding to the common characteristics Sales Order, Customer, and Product are included in one InfoCube:
  - The key figures Sales Order Quantity and Sales Order Price refer to the detail characteristics Sales Order Date and Sales Person.
  - The key figures Delivered Quantity and Delivery Price refer to the detail characteristics Delivery Date and Delivery Person.
  - The key figures Billed Quantity and Billed Price refer to the detail characteristics Billing Date and Accounting Clerk.

Scenario 2: Data Partitioning by a Characteristic in an InfoCube Dimension (account data model)

- One key figure of the data model is defined as InfoObject, which is added to the fact table of the InfoCube.
- The different semantic of this key figure is defined by a characteristic, which builds up a dimension of the InfoCube.

Examples

- The key figure Costs is part of the InfoCube and is uniquely defined by the characteristics Value Type and Version, which build up the dimension Value Type/Version of the InfoCube. The characteristic Value Type can assume the values Plan, Actual, and so on.
- The key figure Total Sales Amount is part of the InfoCube and is uniquely defined by the characteristic Currency Type in a dimension of the InfoCube. The characteristic Currency Type can assume the values Local Currency, Document Currency, and so on.
Scenario 3: Data Partitioning by a MultiCube

Example 1: Plan / Actual data in MultiCube Scenario

- Homogeneous data model
- Some key figures (Plan Costs, Actual Costs) are defined as InfoObjects.
- Each of them will be added to a different basic InfoCube (Plan Data, Actual Data).
- Detailed Views (BW Queries) act on these basic InfoCubes.
- Overview Queries use a MultiCube, which represents the overall reporting scenario (Plan / Actual Data Comparison).

Detailed View: Queries on Plan Data

Basic InfoCube 1
Plan Data
Key Figure: Plan Costs

MultiCube Plan / Actual Data
Key Figures: Plan costs, Actual costs

Detailed View: Queries on Actual Data

Basic InfoCube 2
Actual Data
Key Figure: Actual Costs

Overview Queries:
Plan / Actual Comparison
Scenario 3: Data Partitioning by a MultiProvider

Example 2: Sales Order, Delivery, and Billing Data in a MultiProvider

- Heterogeneous data model (across business scenario): Detailed data of the sub-scenarios (Sales Order, Delivery, Billing) are stored in the corresponding basic InfoCubes. Detailed queries are defined with respect to the basic InfoCubes.

- Overview Queries use a MultiProvider, which represents the overall reporting scenario (Reports on Customer/Product/Sales Order Level).

Valuation of the 3 scenarios

The 3 scenarios described above are compared in the following table, where they are valuated with respect to their possible uses cases.
## Data Partitioning: Comparison of the Scenarios (1)

<table>
<thead>
<tr>
<th>Use Cases for each scenario</th>
<th>Data Model Behavior, if the data granularity varies strongly</th>
<th>Data Model</th>
</tr>
</thead>
</table>
| "Closed" Reporting Scenarios:  
  - Fixed number of key figures  
  - Homogeneous data granularity  
  - All key figures are filled evenly in transaction data. | Characteristics and key figures in the tables of the InfoCubes are distributed sparsely.  
  → Partly empty columns exist  
  This behavior may cause enhanced usage of data storage in the BW system. | Data Model derived from the transaction data model in OLTP:  
  - All key figures defined as in the OLTP system  
  - complex and large fact table  
  - number of dimensions small compared to scenario 2 |

### Scenario 1:
- InfoCube Contains All Key Figures

### Scenario 2:
- Data Partitioning by a Characteristic

### Scenario 3:
- Data Partitioning by a MultiProvider

## Data Partitioning: Comparison of the Scenarios (2)

<table>
<thead>
<tr>
<th>Data Staging in BI</th>
<th>Query Design</th>
<th>Query Performance</th>
</tr>
</thead>
</table>
| Key figures can be transferred 1:1 from OLTP transactional data records into the InfoCube. | Each key figure has clear semantics.  
  The end-user can identify each key figure. | Less complex data access to fact table compared to scenario 2  
  Empty and redundant fields may be selected. |

### Scenario 1:
- InfoCube Contains All Key Figures

### Scenario 2:
- Data Partitioning by a Characteristic

### Scenario 3:
- Data Partitioning by a MultiProvider

### Key Figures
- Key figures can be transferred 1:1 from OLTP transactional data records into the InfoCube.
- Key figures can be transferred 1:1 from OLTP transactional data records into the basic InfoCubes.
- Each sub-scenario has a separate InfoSource for data load.
- parallel data staging is possible.

### Query Design
- Key figures have a clearly defined semantics only in context with a fixed value of the partitioning characteristic.  
  The definition and delivery of restricted key figures is necessary.

### Query Performance
- More complex selection criteria compared to scenario 1  
  because of the use of restricted key figures in BW queries.  
  Selected set of data is smaller compared to scenario 1.  
  Focus queries (such as only Plan data) select on smaller data base compared to scenarios 1 and 2.  
  Overview queries on MultiProvider → parallel data selection for each sub-scenario.
Recommendations

- If you use scenario 3 (MultiProvider), it is important to model the sub-scenarios appropriately. A split into more than 10 sub-scenarios results in unclear data models. Therefore, you can use up to 10 sub-scenarios to take different levels of data granularity into account.

- If you want to read data only from a single basic InfoProvider, use the characteristic 0INFOPROV at query design to filter the basic InfoProvider in scenario 3 (MultiProvider). For more information, see SAP note 728017.

- Often, a combination of the 3 different scenarios leads to a suitable data model in BI.

- If you model completely homogenous data (such as one basic InfoProvider for each fiscal year that is a constant in these basic InfoProviders), use the database partitioning functionality.

Use Case: Data Model for Plan / Actual Comparison with Plan Versions

The reporting scenario contains plan and actual data, which have a different granularity with respect to the time characteristics (Plan data per calendar / fiscal period, actual data per calendar day). Plan data additionally refers to a characteristic Version (for example, defensive variant, and offensive variant). Use a combination of scenario 3 and scenario 2.

- First, use scenario 3 to split the overall data model into two basic InfoCubes (Plan Data, Actual Data).

- Then, use scenario 2 for the plan data:
  The basic InfoCube Plan Data contains one characteristic in a dimension Version. Additional versions loaded from the operational system are mapped by additional values of this characteristic.
3 Data Models Using InfoSets

InfoSets are InfoProviders that logically join data and provide this data for BI queries. InfoSets only reference basic InfoProviders (InfoCubes, DataStore objects, master data InfoObjects), but they contain no data. All the BEx and OLAP services are available (authorizations, texts, variables, hierarchies, calculated key figures) except navigational attributes of InfoSet characteristics. In the InfoSet maintenance, you can make field descriptions unique for the BEx user and hide fields of the basic InfoProviders that are not important for reporting.

When to use InfoSets?

- To join required data from basic InfoProviders
  This allows building a relational BI data model with unified views for reporting (several InfoProviders, but only one view). Therefore, we recommend keeping data in smaller, basic InfoProviders that can be flexibly joined for reporting purposes.
- To allow BEx Reporting on a DataStore object without turning the BEx Reporting indicator on
- To evaluate time dependencies (for example, join time dependent master data InfoObjects)
- To be able to create self joins and left outer joins

Join concepts:

- Inner join: A record can only be in the selected result set if there are entries in both joined tables
- Left outer join: If there is no corresponding record in the right table, the record is part of the result set (fields belonging to the right table have initial values)
- Temporal join: A join is called temporal if at least one member is time-dependent.
- Self join: The same object is joined together

3.1 Second-Order Navigational Attributes

Use case

- An InfoProvider contains characteristics that have navigational attributes. These are the first-order navigational attributes of the InfoProvider.
- These first order navigational attributes again have navigational attributes. These are the second-order navigational attributes of the InfoProvider.
- The end-user wants to navigate / drill-down to these second-order navigational attributes in BI queries based on the InfoProvider.
Example: Transaction Data Using Consolidated InfoObjects

1. DataStore object Purchase Order (ZPUR_O01) contains consolidated InfoObjects 0PRODUCT and 0GN_VENDOR.

2. Consolidated InfoObjects 0PRODUCT and 0GN_VENDOR contain local-view InfoObjects 0MATERIAL (R/3) / 0BBP_PROD (SRM) and 0VENDOR (R/3) / 0BPARTNER (SRM) as first-order navigational attributes.

3. Local-View InfoObjects 0MATERIAL (R/3) / 0BBP_PROD (SRM) and 0VENDOR (R/3) / 0BPARTNER (SRM) contain local-view attributes as second-order navigational attributes with respect to the DataStore object Purchase Order (ZPUR_O01).

4. In queries based on DataStore object Purchase Order (ZPUR_O01), the user wants to navigate / drill-down to the second-order navigational attributes 0MATL_GROUP, 0PROD_TYPE, 0INDUSTRY, 0IND_CODE.

Excursion: Why do consolidated InfoObjects need second-order navigational attributes?

- One basic idea of the consolidated InfoObjects is to keep these InfoObjects as lean as possible; that is, only attributes used for global (cross source systems and applications) reporting are relevant.
- The consolidated InfoObject cannot contain any attribute of the local views.
- Global attributes delivered by SAP can only be:
  - Consolidated InfoObjects
  - InfoObjects with unified or standardized values in all source systems
  - DUNS number, ISO Codes, UNSPSC Codes, and so on.
Solution 1: Use InfoSet to Build a Star Schema

- Create an InfoSet starting with the InfoProvider.
- Join the master data tables of the characteristics, which lead to the second-order navigational attributes, to the InfoProvider.
- Join the master data tables of the first-order navigational attributes, which contain the second-order navigational attributes, to the master data tables of the characteristics.
- Define queries based on the InfoSet using the second-order navigational attributes for navigation / drill down.

Example:

```
0MATERIAL ... 0MATL_GROUP ...
0PRODUCT ... 0MATERIAL 0BBP_PROD ...
0BBP_PROD ...
PROD_TYPE ...
0PRODUCT ...
0GN_VENDOR ...
QUANTITY VALUE
POGUID ...
0MATERIAL 0GN_VENDOR ...
0VENDOR 0BPARTNER ...
0VENDOR ...
0INDUSTRY ...
0BPARTNER ...
0IND_CODE ...
```

Arguments pro and contra this solution (the use of an InfoSet to model second-order navigational attributes)

Pro:
- All components of the data model (InfoProvider, InfoObjects) can be kept lean. There are no data redundancies. No additional master data attributes have to be determined during extraction or upload of transaction data.
- Clear, intuitive, and flexible data model: You can add additional second-order navigational attributes by extension of the InfoSet. Therefore, no additional data load is required after a data model extension.

Contra:
- Performance problems may occur at BI query runtime if too many master data table of first- and second-order navigational attributes are joined to the InfoProvider.

There are other solutions without the use of InfoSets to model second-order navigational attributes.
- Solution 2 (Extension of the InfoProvider): Add the first-order navigational attributes, which contain the second-order navigational attributes, to the characteristics of the InfoProvider (in addition to the existing characteristics).
- Solution 3 (Extension of the characteristics): Add the second-order navigational attributes, as navigational attributes to the characteristics of the InfoProvider.
Solutions 2 and 3 have better performance at BI query runtime, if the InfoProvider is an InfoCube: The relationship between first- and second-order navigational attributes is fixed by the BI data model and data is persistently loaded to InfoProviders (solution 2) or master data tables (solution 3).
The BI data model of solutions 2 and 3 cause

- redundant data in BI
- inflexible data model: Additional characteristics (such as first-order navigational attributes) in the InfoProvider cause a reload of transactional data
- realignment problems within the InfoProvider if the assignment of InfoObjects to first-order navigational attributes changes

For each use case, where second-order navigational attributes are needed, determine if the BI data model has to be flexible with regard to changes (solution 1) or if the BI query performance has to be optimized (solution 2 or 3).

3.2 Data Model Examples for Temporal Joins

Example 1: Two time-dependent master data InfoObjects are joined.

The InfoObject, Cost Center (technical name CSTCNTR) is joined to a second InfoObject, Profit Center (technical name PROFITC). InfoObject, Cost Center, has the attribute Profit Center, which is used as join condition. Both InfoObjects are time-dependent and contain the attribute, Responsible Person (technical name RESPPERS).
The temporal join gives insight to the question: Which person responsible for a cost center has worked together with a certain person responsible for a profit center at the same time?

Master data table of InfoObject Cost Center:

<table>
<thead>
<tr>
<th>CSTCNTR (key)</th>
<th>DATETO (key)</th>
<th>DATEFROM</th>
<th>PROFITC</th>
<th>RESPPERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4711</td>
<td>31.05.2001</td>
<td>01.01.2001</td>
<td>BI</td>
<td>Jack</td>
</tr>
<tr>
<td>4711</td>
<td>31.12.2001</td>
<td>01.06.2001</td>
<td>BI</td>
<td>John</td>
</tr>
<tr>
<td>4711</td>
<td>31.12.9999</td>
<td>01.01.2002</td>
<td>BI</td>
<td>Joe</td>
</tr>
<tr>
<td>0815</td>
<td>31.01.2001</td>
<td>01.01.2001</td>
<td>KM</td>
<td>Jane</td>
</tr>
<tr>
<td>0815</td>
<td>31.12.9999</td>
<td>01.02.2001</td>
<td>KM</td>
<td>Jill</td>
</tr>
</tbody>
</table>

Master data table of InfoObject Profit Center:

<table>
<thead>
<tr>
<th>PROFITC (key)</th>
<th>DATETO (key)</th>
<th>DATEFROM</th>
<th>RESPPERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI</td>
<td>30.06.2001</td>
<td>01.01.2000</td>
<td>Karl</td>
</tr>
<tr>
<td>BI</td>
<td>31.12.9999</td>
<td>01.07.2001</td>
<td>Hanna</td>
</tr>
</tbody>
</table>

The InfoSet joins both tables using the condition: PROFITC of master data table CSTCNTR is equal to the PROFITC of master data table PROFITC.

A query uses this InfoSet and selects Cost Center “4711”. Without consideration of any time-dependencies, the result of the query includes 6 records. For simplicity, the columns CSTCNTR = “4711” (query selection) and PROFITC = “BI” (join condition) are left out:

<table>
<thead>
<tr>
<th>RESPPERS (CSTCNTR)</th>
<th>RESPPERS (PROFITC)</th>
<th>DATEFROM (CSTCNTR)</th>
<th>DATETO (CSTCNTR)</th>
<th>DATEFROM (PROFITC)</th>
<th>DATETO (PROFITC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack</td>
<td>Karl</td>
<td>01.01.2001</td>
<td>31.05.2001</td>
<td>01.01.2000</td>
<td>30.06.2001</td>
</tr>
<tr>
<td>John</td>
<td>Karl</td>
<td>01.06.2001</td>
<td>31.12.2001</td>
<td>01.01.2000</td>
<td>30.06.2001</td>
</tr>
</tbody>
</table>
The result in consideration of time-dependencies (only 4 valid records) is shown in the following figure.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack</td>
<td>Karl</td>
<td>01.01.2001</td>
<td>31.05.2001</td>
<td>01.01.2000</td>
<td>30.06.2001</td>
</tr>
<tr>
<td>John</td>
<td>Karl</td>
<td>01.06.2001</td>
<td>31.12.2001</td>
<td>01.01.2000</td>
<td>30.06.2001</td>
</tr>
<tr>
<td>Joe</td>
<td>Karl</td>
<td>01.01.2002</td>
<td>31.12.9999</td>
<td>01.01.2000</td>
<td>30.06.2001</td>
</tr>
<tr>
<td>Jack</td>
<td>Hanna</td>
<td>01.01.2001</td>
<td>31.05.2001</td>
<td>01.07.2001</td>
<td>31.12.9999</td>
</tr>
<tr>
<td>Joe</td>
<td>Hanna</td>
<td>01.01.2002</td>
<td>31.12.9999</td>
<td>01.07.2001</td>
<td>31.12.9999</td>
</tr>
</tbody>
</table>

The two records marked in red do not have overlapping time intervals [DATEFROM; DATETO] for Cost Center and Profit Center master data tables. Therefore, these records are not selected by the temporal join. The other four records have an overlapping time interval and are returned to the BI query. The boundaries DATEFROM and DATETO of valid time interval for each record are marked by green fields and are also displayed in a time bar diagram below.

**Example 2: Time-dependent master data InfoObject is joined to a DataStore object containing a key date.**

The InfoObject, Cost Center (technical name CSTCNTR) is joined to a DataStore object, Invoice (technical name INVOICE). The DataStore object has the characteristic Cost Center, which is used as a join condition. The InfoObject CSTCNTR is time dependent, whereas the DataStore object INVOICE has the InfoObject calendar day (technical name 0CALDAY), which can be used as key date. The temporal join gives insight to the question: Who was the responsible person of a Cost Center at the date when the invoice was posted?
Master data table of InfoObject Cost Center:

<table>
<thead>
<tr>
<th>CSTCNTR (key)</th>
<th>DATETO (key)</th>
<th>DATEFROM</th>
<th>RESPPERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4711</td>
<td>31.05.2001</td>
<td>01.01.2001</td>
<td>Jack</td>
</tr>
<tr>
<td>4711</td>
<td>31.12.2001</td>
<td>01.06.2001</td>
<td>John</td>
</tr>
<tr>
<td>4711</td>
<td>31.12.9999</td>
<td>01.01.2002</td>
<td>Joe</td>
</tr>
<tr>
<td>0815</td>
<td>31.01.2001</td>
<td>01.01.2001</td>
<td>Jane</td>
</tr>
<tr>
<td>0815</td>
<td>31.12.9999</td>
<td>01.02.2001</td>
<td>Jill</td>
</tr>
</tbody>
</table>

DataStore Object INVOICE:

<table>
<thead>
<tr>
<th>INVOICE ID (key)</th>
<th>0CALDAY</th>
<th>CSTCNTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>100001</td>
<td>03.02.2001</td>
<td>4711</td>
</tr>
<tr>
<td>100002</td>
<td>28.04.2002</td>
<td>4711</td>
</tr>
</tbody>
</table>

The InfoSet joins both tables using the condition:
CSTCNTR of master data table CSTCNTR is equal to the CSTCNTR of DataStore Object INVOICE.

The result in consideration of time-dependencies (only 2 valid records) is shown in the following figure.
A record of the InfoSet is selected by the BI query only if the key date of the DataStore object INVOICE is within the valid time interval [DATEFROM; DATETO] of the master data InfoObject CSTCNTR.

The derivation of a key date is also possible for the time characteristics of the SAP BI system:

- 0CALWEEK  Calendar Year / Week
- 0CALMONTH  Calendar Year / Month
- 0CALQUARTER  Calendar Year / Quarter
- 0CALYEAR  Calendar Year
- 0FISCPER  Fiscal Year / Period (compound with fiscal year variant 0FISCVARNT)
- 0FISCYEAR  Fiscal Year (compound with the fiscal year variant 0FISCVARNT)

You can choose the begin date or end date of the above time intervals or a fixed date within these intervals.

**Example 3: Time-dependent master data InfoObject is joined to a DataStore object, where a valid time interval can be derived from a time characteristic.**

The InfoObject, Cost Center (technical name CSTCNTR) is joined to a DataStore object, Project Sponsor (technical name SPONSOR). The DataStore object has the characteristic Cost Center, which is used as join condition. The InfoObject CSTCNTR is time dependent, whereas the DataStore object SPONSOR has the key field Calendar Month (technical name 0CALMONTH) to define the valid time interval of each record. The temporal join gives insight to the question: Who was the responsible person of a Cost Center in the time interval when a project of the Cost Center was sponsored?

Master data table of InfoObject Cost Center:

<table>
<thead>
<tr>
<th>CSTCNTR (key)</th>
<th>DATETO (key)</th>
<th>DATEFROM</th>
<th>RESPPERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4711</td>
<td>31.05.2001</td>
<td>01.01.2001</td>
<td>Jack</td>
</tr>
<tr>
<td>4711</td>
<td>31.12.2001</td>
<td>01.06.2001</td>
<td>John</td>
</tr>
<tr>
<td>4711</td>
<td>31.12.9999</td>
<td>01.01.2002</td>
<td>Joe</td>
</tr>
<tr>
<td>0815</td>
<td>31.01.2001</td>
<td>01.01.2001</td>
<td>Jane</td>
</tr>
<tr>
<td>0815</td>
<td>31.12.9999</td>
<td>01.02.2001</td>
<td>Jill</td>
</tr>
</tbody>
</table>

DataStore Object SPONSOR:

<table>
<thead>
<tr>
<th>SPONSOR (key)</th>
<th>0CALMONTH (key)</th>
<th>CSTCNTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hugo</td>
<td>04.2003</td>
<td>4711</td>
</tr>
<tr>
<td>Alice</td>
<td>02.2001</td>
<td>4711</td>
</tr>
</tbody>
</table>

The InfoSet joins both tables using the condition: CSTCNTR of master data table CSTCNTR is equal to the CSTCNTR of DataStore Object SPONSOR.

The time interval [DATEFROM; DATETO] for each record in the DataStore Object SPONSOR can now be derived from the time characteristic 0CALMONTH:

<table>
<thead>
<tr>
<th>0CALMONTH (Source)</th>
<th>DATEFROM (Target)</th>
<th>DATETO (Target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>02.2001</td>
<td>01.02.2001</td>
<td>28.02.2001</td>
</tr>
</tbody>
</table>
Similarly, you can retrieve the valid time interval \([\text{DATEFROM}; \text{DATETO}]\) for the time characteristics of the SAP BI system:

- 0CALWEEK Calendar Year / Week
- 0CALMONTH Calendar Year / Month
- 0CALQUARTER Calendar Year / Quarter
- 0CALYEAR Calendar Year
- 0FISCPER Fiscal Year / Period (compound with the fiscal year variant 0FISCVARNT)
- 0FISCYEAR Fiscal Year (compound with the fiscal year variant 0FISCVARNT)

The result in consideration of time-dependencies (only 2 valid records) is shown in the following figure.

### Result in Consideration of Time Dependencies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack</td>
<td>Alice</td>
<td>01.01.2001</td>
<td>31.05.2001</td>
<td>01.02.2001</td>
<td>28.02.2001</td>
</tr>
<tr>
<td>John</td>
<td>Alice</td>
<td>01.06.2001</td>
<td>31.12.2001</td>
<td>01.02.2001</td>
<td>28.02.2001</td>
</tr>
<tr>
<td>Joe</td>
<td>Alice</td>
<td>01.01.2002</td>
<td>31.12.9999</td>
<td>01.02.2001</td>
<td>28.02.2001</td>
</tr>
<tr>
<td>Jack</td>
<td>Hugo</td>
<td>01.01.2001</td>
<td>31.05.2001</td>
<td>01.04.2003</td>
<td>30.04.2003</td>
</tr>
</tbody>
</table>

The four records marked in red do not have overlapping time intervals \([\text{DATEFROM}; \text{DATETO}]\) for Cost Center master data tables and DataStore Object Sponsor. The other two records have an overlapping time interval and are returned to the query. The boundaries DATEFROM and DATETO of valid time interval for each record are marked by green fields and are also displayed in a time bar diagram.
4 Performance Aspects of MultiProviders and InfoSets

**MultiProviders**

- These are the advantages of MultiProviders:
  - Local queries (on each basic InfoProvider) versus global queries (on MultiProvider, parallel execution).
  - Independent (and parallel) data load into the basic InfoProviders.
  - Small total data volumes: basic InfoProvider have less redundant data; they are sparsely filled and less complex.
- As a general rule, we recommend assigning up to 10 basic InfoProviders to a MultiProvider. If the number of basic InfoProviders is significantly higher, the overhead in combining results at BI query runtime may become excessive.
- If a MultiProvider contains at least one basic InfoProvider with non-cumulative key figures, all queries are processed sequentially.
- The performance optimizing tools of the OLAP (such as caching, aggregation) only work for a MultiProvider if all constituent basic InfoProviders of the MultiProvider support these tools.

**InfoSets**

- InfoSets do not have the set of performance tools as InfoCubes (such as aggregates, partitioning, and compression).
- Use left outer joins in InfoSets only when necessary. A left outer join has a significantly worse performance than a corresponding inner join.
- If your reporting requirements on a DataStore Object are very restricted (that is, you want to display only very few, selective records), use an InfoSet on top of the DataStore object and disable the BEx Reporting indicator. This results in better data loading performance, but also in worse performance at BI query runtime if more than 10 records are selected from the DataStore Object.

5 List of Documents Related to MultiProviders and InfoSets

**MultiProviders**: For more information, see the SDN document How to Create Efficient MultiProvider Queries [https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/751be690-0201-0010-5e80-f4f92f4e4a4](https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/751be690-0201-0010-5e80-f4f92f4e4a4)

**InfoSets**: For more information, see following OSS notes:

- 583249: Temporal joins
- 592785: Interpretation of results
- 577953: Left outer joins