Analysis Process Modeling Notation For Business Intelligence (APMN4BI)

Thomas Neuböck

October 2017

Contents

1	Pre	limina	ries for APMN4BI	1
	1.1	Runni	ng Example	1
	1.2	Enrich	aed Dimensional Fact Model (eDFM)	1
2	Ana	alysis S	Situations	7
	2.1	Non-co	omparative Analysis Situations	7
	2.2	Compa	arative Analysis Situations	7
3	Nav	vigation	n Operators	17
	3.1	Naviga	ation Step	17
	3.2	Opera	tors Not Involving Comparison	17
		3.2.1	Operators Changing Granularity Level	17
		3.2.2	Operators Changing Dice Node	17
		3.2.3	Operators Changing Slice Conditions	17
		3.2.4	Operators Changing Base Measure Conditions	17
		3.2.5	Operators Changing Measures	24
		3.2.6	Operators Changing Filters	24
		3.2.7	Operators Changing Cube Access	24
	3.3	Opera	tors Involving Comparison	24
		3.3.1	Operators Introducing Comparison	24
		3.3.2	Operators Changing Comparison	27
		3.3.3	Operators Dropping Comparison	27
	3.4	Use of	Analysis Situations as Cubes	27

ii	CO.	NTENTS
ii	CO.	NTENTS

4	Ext	ension to Schema Level	37
	4.1	Analysis Situation Schema	37
	4.2	Navigation Step Schema	37
	4.3	Navigation guards	37
5	Bus	siness Intelligence (BI) Analysis Graphs	49
	5.1	Definition of BI Analysis Graphs	49
	5.2	BI Analysis Graph Schemas	49
	5.3	Instances of Analysis Graph Schemas	49
	5.4	Composite analysis situation	49
Li	st of	Figures	49
Li	st of	Tables	56

Chapter 1

Preliminaries for APMN4BI

1.1 Running Example

see Figure 1.1

1.2 Enriched Dimensional Fact Model (eDFM)

see Figure 1.2, 1.3, 1.4

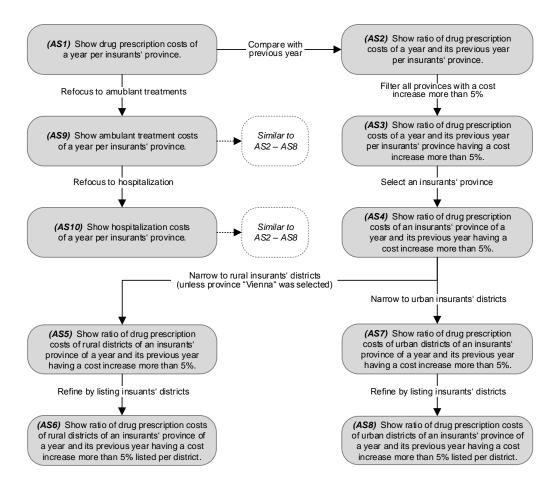


Figure 1.1: Health care use case – overview of an analysis process

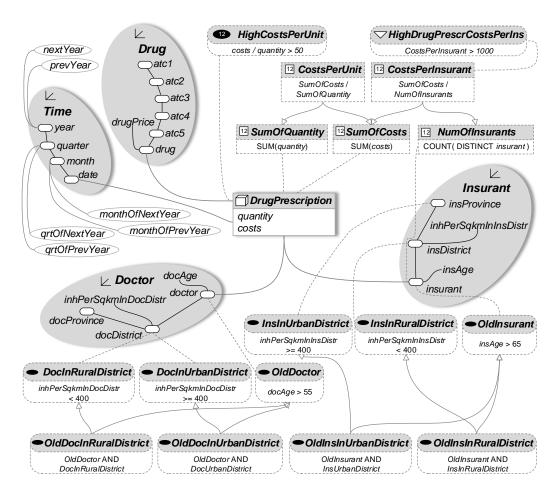


Figure 1.2: eDFM for Drug Prescriptions

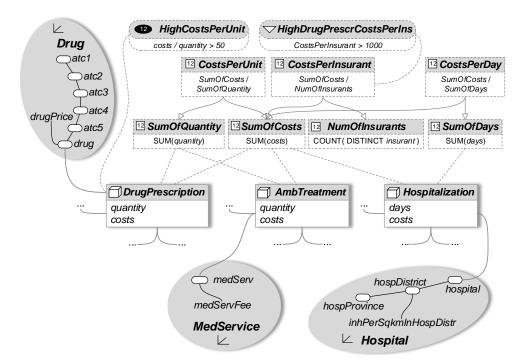
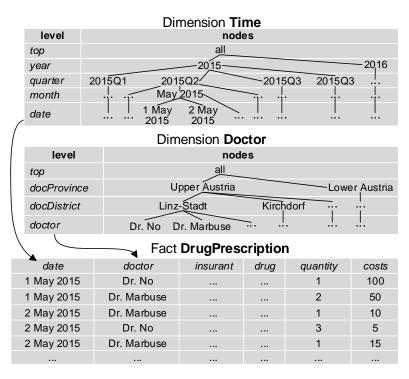


Figure 1.3: eDFM for Ambulant Treatment and Hospitalization



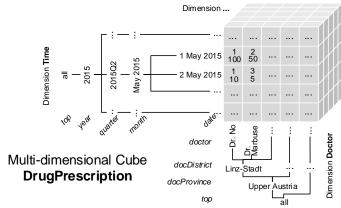


Figure 1.4: Cube Instance

Chapter 2

Analysis Situations

2.1 Non-comparative Analysis Situations

see Figure 2.1, 2.2, 2.3, 2.4, 2.5

2.2 Comparative Analysis Situations

see Figure 2.6, 2.7, 2.8, 2.9, 2.10

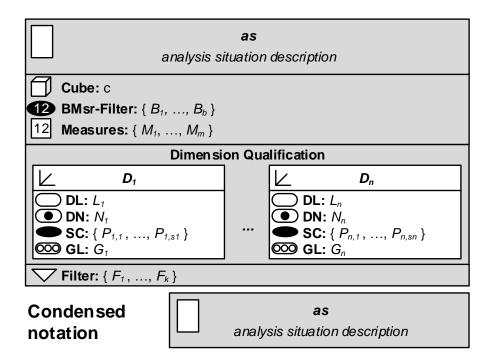


Figure 2.1: Graphical Template of a Non-comparative Analysis Situation

HighDrugPrescrCostsInRuralDistrPerProv High drug prescription costs in rural insurants' districts in year 2016 listed per insurants' province having high drug prescription costs per insurant.							
Cube: DrugPrescription 12 BMsr-Filter: { HighCostsPerUnit } 12 Measures: { SumOfCosts, CostsPerInsurant }							
	Dimension Qualific	ation					
	<u> </u>	<u></u>	∠ Doctor				
□ DL: year □ DL: top □ DL: top □ DL: top □ DL: top □ DN: all □ DN: all							
Filter: { Hig	hDrugPrescrCostsPerIns }						

Figure 2.2: Example of a Graphical Representation of a Non-comparative Analysis Situation

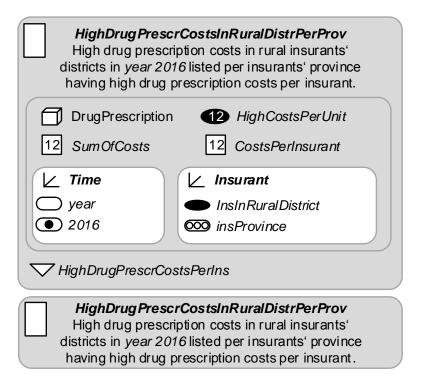


Figure 2.3: Example of an Alternative Graphical Representation of a Non-comparative Analysis Situation

SQL translation:

select $G_1, ..., G_n$,

 $M_1, \ldots, M_m,$

from c natural join D₁ natural join ... natural join D_n

where B_1 and ... and B_b and

 $L_1 = N_1$ and ... and $L_n = N_n$ and $P_{1,1}$ and ... and $P_{1,s1}$ and ...

 $P_{n,1}$ and ... and $P_{n,sn}$

group by $G_1, ..., G_n$

having F_1 and ... and F_k

Result set:

G ₁	 G _n	M ₁	 M _m

Figure 2.4: SQL Translation

SQL translation:

select insProvince,

SUM(costs) as SumOfCosts, SUM(costs) / COUNT(DISTINCT insurant) as CostsPerInsurant

from DrugPrescription natural join

Time natural join

Insurant

where costs / quantity > 50 and

year = 2015 and

inhPerSqkmInInsDistr < 400

group by insProvince

having SUM(costs) / COUNT(DISTINCT insurant) > 1000

Example result set:

insProvin <i>c</i> e	SumOfCosts	Costs PerInsurant
Upper Austria	117790612.01	2120.31
Lower Austria	197790612.45	2415.07
		•••

Figure 2.5: Example of an SQL Translation

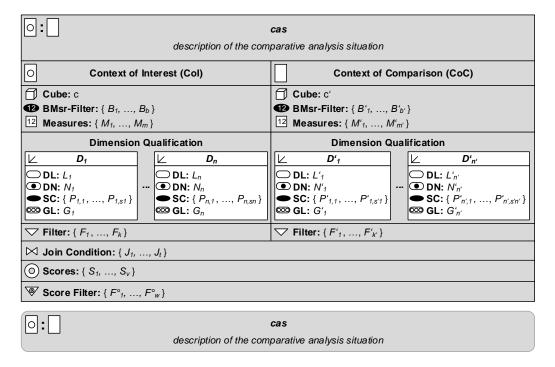


Figure 2.6: Graphical Template of a Comparative Analysis Situation

High Drug Prescr CostsIn Rural	High Drug Prescr CostsIn Rural Districts Compared With Prev Year
Increases of high drug prescription of 2016 compared with previous yea	Increases of high drug prescription costs in rural insurants' districts in year 2016 compared with previous year 2015 listed per insurants' province
O Context of Interest (Col)	Context of Comparison (CoC)
☐ Cube: DrugPrescription ☐ BMsr-Filter: { HighCostsPerUnit } ☐ Measures: { SumOfCosts, CostsPerInsurant }	☐ Cube: DrugPrescription ⚠ BMsr-Filter: { HighCostsPerUnit } ☐ Measures: { SumOfCosts, CostsPerInsurant }
Dimension Qualification	Dimension Qualification
L Time L Insurant L Drug L Doctor ○ DL: year ○ DL: top ○ DL: top ○ DL: top ● DN: 2016 ● DN: all ● DN: all ● DN: all ● SC: { InshRuralDistrict } ● SC: { } ● SC: { } ● GL: top ● GL: top ● GL: top	L Time L Insurant L Drug L Doctor ○ DL: year ○ DL: top ○ DL: top ○ DL: top ⑤ DN: 2015 ⑥ DN: all ⑥ DN: all ⑥ DN: all ⑥ SC: { InsInRuralDistrict } ⑥ SC: { InsInRuralDistrict } ⑥ SC: { InsInRuralDistrict }
\boxtimes Join Condition: { $Col.insProvince = Coc.insProvince, Col.top_{Time} = Coc.top_{Time}, Col.top_{Drug} = Coc.top_{Dug.}$	$T_{IMP} = CoC.$ $top_{TIMP}, Col.$ $top_{Drug} = CoC.$ $top_{Drug},$
O Scores: { RatioOfSumOfCosts, RatioOfCostsPerInsurant }	
▼ Score Filter: { IncreasedCostsPerInsurant }	
High Drug Pres cr Cost sIn Rural I	High Drug Prescr Costs In Rural Districts Compared With Prev Year
1	Increases of high drug prescription costs in rural insurants' districts in year 2016 compared with previous year 2015 listed per insurants' province

Figure 2.7: Example of a Graphical Representation of a Comparative Analysis Situation

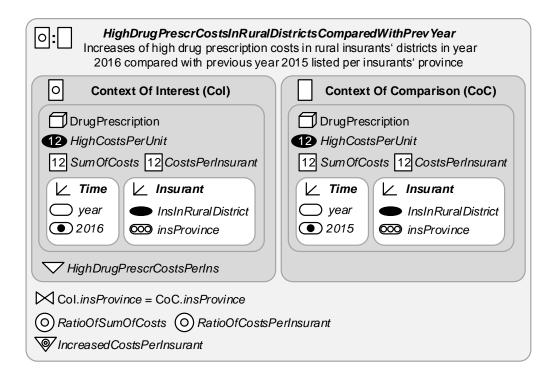


Figure 2.8: Example of an Alternative Graphical Representation of a Comparative Analysis Situation

SQL translation:

select		II. G_n , Col. M_1 ,, Col. M_m , CoC. G'_n , CoC. M'_1 ,, CoC. M'_m ,	
from	(select from where group by having) Col	$G_1, \ldots, G_n, M_1, \ldots, M_m$ c natural join D_1 natural join \ldots B_1 and \ldots and B_b $L_1 = N_1$ and \ldots and $L_n = N_n$ $P_{1,1}$ and \ldots and P_{1,s_1} \ldots $P_{n,1}$ and \ldots and P_{n,s_n} G_1, \ldots, G_n F_1 and \ldots and F_k	natural join D _n and and and and
inner join	(select from where group by having) CoC	$G_{1}^{\prime}, \ldots, G_{n}^{\prime}, M_{1}^{\prime}, \ldots, M_{m}^{\prime}$ contaural join D_{1}^{\prime} natural join D_{1}^{\prime} and D	. natural join Dʻ _n and and and and
on	J_1 and and	J_t	
where	F°₁ and and	i F° _w	

Result set:

Col.G₁	 Col.G _n	Col.M₁	 Col.M _n	CoC.Gʻ ₁	 CoC.Gʻnʻ	CoC.M'1	 CoC.M'n	Sı	 St
	 		 		 		 	:	

Figure 2.9: SQL Translation of a Comparative Analysis Situation

SQL translation:

Col.insProvince, Col.Sum OfCosts, Col.CostsPerInsurant, select

CoC.insProvince, CoC.SumOfCosts, CoC.CostsPerInsurant, Col.SumOfCosts / CoC.SumOfCosts as RatioOfS as RatioOfSumOfCosts, Col.CostsPerInsurant / CoC.CostsPerInsurant as RatioOfCostsPerInsurant

 $ins Province, SUM(costs) \ as \ SumOfCosts, SUM(costs) \ / \ COUNT(\ DISTINCT\ insurant) \ as \ Costs Perlnsurant \ Drug Prescription\ natural\ join\ \textit{Time}\ natural\ join\ \textit{Insurant}$ from (select

from

where costs / quantity > 50 and

year = 2016 and inhPerSqkmInInsDistr < 400

group by

insProvince SUM(costs) / COUNT(DISTINCT insurant) > 1000 having) Col

 $insProvince, SUM(costs) \ as \ SumOfCosts, SUM(costs) / \ COUNT(\ DISTINCT\ insurant) \ as \ CostsPerInsurant \ DrugPrescription\ natural\ join\ Time\ natural\ join\ Insurant$ inner join (select

from

where costs / quantity > 50 and

year = 2015 and inhPerSqkmInInsDistr < 400

group by) CoC insProvince

on Col.insProvince = CoC.insProvince

Col.CostsPerInsurant / CoC.CostsPerInsurant > 1 where

Example result set:

Col.	Col.	Col.	CoC.	CoC.	CoC.	RatioOfSumOf	RatioOfCostsPer
insProvince	SumOfCosts 5 4 1	Costs PerInsurant	insProvince	SumOfCosts	CostsPerInsurant	Costs	Insurant
Upper Austria	117790612.01	2120.31	Upper Austria	107790612.01	1959.21	1,0928	1.0822
Lower Austria	197790612.45	2415.07	Lower Austria	187790612.45	2295.74	1.0533	1.0520

Figure 2.10: Example of an SQL Translation of a Comparative Analysis Situation

Chapter 3

Navigation Operators

3.1 Navigation Step

see Figure 3.1

3.2 Operators Not Involving Comparison

3.2.1 Operators Changing Granularity Level

see Table 3.1

3.2.2 Operators Changing Dice Node

see Table 3.2, 3.3

3.2.3 Operators Changing Slice Conditions

see Table 3.4

3.2.4 Operators Changing Base Measure Conditions

see Table 3.5

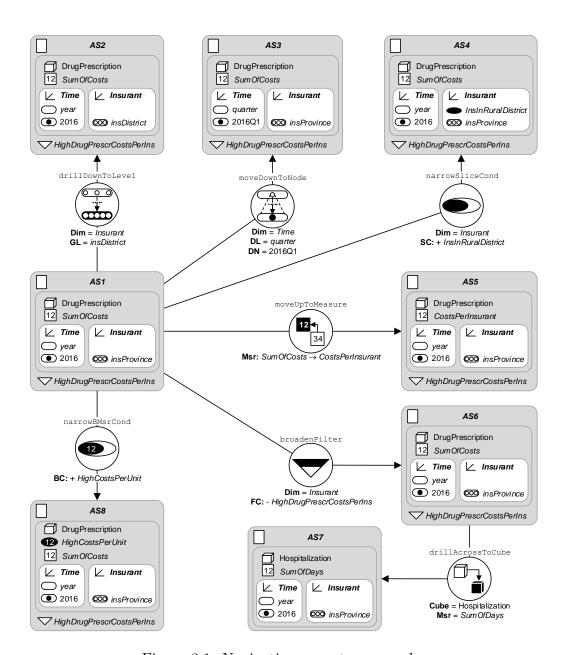


Figure 3.1: Navigation operator examples

Table 3.1: Operators drillDownOneLevel, drillDownToLevel, rollUpOneLevel, and rollUpToLevel

Operator Definition	Symbol
$\frac{\texttt{drillDownOneLevel}(D)}{}$	drillDownOneLevel
Precondition: $GranLvl_{src}(D) \neq base_D$	000
Postcondition: $GranLvl_{trg}(D) \rightarrow GranLvl_{src}(D)$	
	$\mathbf{Dim} = D$
$\underline{\mathtt{drillDownToLevel}(D,G)}$	drillDownToLevel
Precondition: $G \rightarrow GranLvl_{src}(D)$	
Postcondition: $GranLvl_{trg}(D) = G$	
	$\mathbf{Dim} = D$
	$\mathbf{GL} = G$
rollUpOneLevel(D)	rollUpOneLevel
Precondition: $GranLvl_{src}(D) \neq top_D$	
Postcondition: $GranLvl_{src}(D) \rightarrow GranLvl_{src}(D)$	00000
	$\mathbf{Dim} = D$
rollUpToLevel(D,G)	rollUpToLevel
Precondition: $GranLvl_{src}(D) \twoheadrightarrow G$	
Postcondition: $GranLvl_{trg}(D) = G$	00000
	$\mathbf{Dim} = D$
	$\mathbf{GL} = G$

 $Table \ 3.2: \ Operators \ {\tt moveDownToNode}, \ {\tt moveUpToNode}, \ {\tt moveAsideToNode},$

and moveToNode

Operator Definition	Symbol
$\boxed{ \texttt{moveDownToNode}(D, L, N) }$	moveDownToNode
Precondition: $L \in Lvls_D, L \neq base_D,$	
$L \rightarrow DiceLvl_{src}(D), N \in Nodes_d(L)$ with	
$d = DimInstance_{src}(D)$, and $N \rightarrow DiceNode_{src}(D)$	$\mathbf{Dim} = D$
Postcondition:	$\mathbf{DL} = L$
$L = DiceLvl_{trg}(D), N = DiceNode_{trg}(D)$	$\mathbf{DN} = N$
${\tt moveUpToNode}(D, L)$	
Precondition: $L \in Lvls_D, L \neq top_D,$	moveUpToNode
$DiceLvl_{src}(D) woheadrightarrow L, DiceNode_{src}(D) woheadrightarrow N$	
for the unique existing $N \in Nodes_d(L)$, with	
$d = DimInstance_{src}(D)$	D:
Postcondition:	$\mathbf{Dim} = D$
$L = DiceLvl_{trg}(D), N = DiceNode_{trg}(D)$	$\mathbf{DL} = L$
${\tt moveAsideToNode}(D,N)$	moveAsideToNode
Precondition: $DiceLvl_{src}(D) \neq top_D$,	A
$N \neq DiceNode_{src}(D)$ and, for the unique	
existing \hat{N} with $DiceNode_{src}(D) \to \hat{N}: N \to \hat{N}$	$\widetilde{\mathbf{Dim}} = D$
Postcondition: $DiceNode_{trg}(D) = N$	$\mathbf{DN} = N$
moveToNode(D, L, N)	moveToNode
Precondition: $L \neq DiceLvl_{src}(D), L \in Lvls_D,$	
and $N \in Nodes_d(L)$, with $d = DimInstance_{src}(D)$	
Postcondition: $DiceLvl_{trg}(D) = L$,	$\mathbf{Dim} = D$
$DiceNode_{trg}(D) = N$	$\mathbf{DL} = L$
	$\mathbf{DN} = N$

 $\begin{tabular}{ll} Table 3.3: Operators \verb|moveDownToFirstNode|, moveDownToLastNode|, moveToNextNode|, and \verb|moveToPrevNode| \\ \end{tabular}$

Operator Definition	Symbol
$\boxed{ \texttt{moveDownToFirstNode}(D) }$	moveDownToFirstNode
Precondition: $DiceLvl_{src}(D) \neq base_D$	moveDownToFITstNode
Postcondition: $DiceLvl_{trg}(D) \rightarrow DiceLvl_{src}(D)$,	
$DiceNode_{trg}(D) =$	D:
$min_{\prec}\{N\mid N \to DiceNode_{src}(D)\}$	$\mathbf{Dim} = D$
$\underline{\texttt{moveDownToLastNode}(D)}$	moveDownToLastNode
Precondition: $DiceLvl_{src}(D) \neq base_D$	moveBownToEdstrode
Postcondition: $DiceLvl_{trg}(D) \rightarrow DiceLvl_{src}(D)$,	
$DiceNode_{trg}(D) =$	Diag. D
$max_{\prec}\{N \mid N \to DiceNode_{src}(D)\}$	$\mathbf{Dim} = D$
$\underline{\mathtt{moveToNextNode}(D)}$	
Precondition: $DiceLvl_{src}(D) \neq top_D$ and, for	moveToNextNode
the unique existing N with $DiceNode_{src}(D) \to N$,	moveroweakwade
there exits an $N' \to N$, such that	
$N' = next_{\prec}(DiceNode_{src}(D))$	D:
Postcondition: $DiceNode_{trg}(D) = N'$	$\mathbf{Dim} = D$
$\underline{\texttt{moveToPrevNode}(D)}$	
Precondition: $DiceLvl_{src}(D) \neq top_D$ and, for	moveToPrevNode
the unique existing N with $DiceNode_{src}(D) \to N$,	movelorievnode
there exits an $N' \to N$, such that	
$N' = prev_{\prec}(DiceNode_{src}(D))$	
Postcondition: $DiceNode_{trg}(D) = N'$	$\mathbf{Dim} = D$

 ${\bf Table~3.4:~Operators~narrowSliceCond,~broadenSliceCond,~and~refocus-}$ SliceCond

Operator Definition	Symbol
$narrowSliceCond(D, +P_1, \cdots, +P_s)$	narrowSliceCond
Precondition: $P_1, \dots, P_s \in DimPredicates_D$,	
$\{P_1, \cdots, P_s\} \nsubseteq SliceConds_{src}(D)$	
Postcondition: $SliceConds_{trg}(D) =$	$\widetilde{\mathbf{Dim}} = D$
$SliceConds_{src}(D) \cup \{P_1, \cdots, P_s\}$	$\mathbf{SC}\colon +P_1,\cdots,+P_s$
$narrowSliceCond(D, P_{old} \rightarrow P_{new})$	
Precondition: $P_{old}, P_{new} \in DimPredicates_D,$	narrowSliceCond
$P_{old} \in SliceConds_{src}(D),$	
$P_{new} \notin SliceConds_{src}(D), P_{new} \Rightarrow P_{old}$	
Postcondition: $SliceConds_{trg}(D) =$	$\mathbf{Dim} = D$
$SliceConds_{src}(D) \setminus \{P_{old}\} \cup \{P_{new}\}$	$\mathbf{SC} \colon P_{old} \to P_{new}$
$\underline{\mathtt{broadenSliceCond}(D, -P_1, \cdots, -P_s)}$	broadenSliceCond
Precondition: $P_1, \dots, P_s \in DimPredicates_D$,	
$\{P_1, \cdots, P_s\} \nsubseteq SliceConds_{src}(D)$	
Postcondition: $SliceConds_{trg}(D) =$	$\mathbf{Dim} = D$
$SliceConds_{src}(D)\backslash\{P_1,\cdots,P_s\}$	\mathbf{SC} : $-P_1, \cdots, -P_s$
$\underline{\text{broadenSliceCond}(D, P_{old} \rightarrow P_{new})}$	
Precondition: $P_{old}, P_{new} \in DimPredicates_D,$	broadenSliceCond
$P_{old} \in SliceConds_{src}(D),$	
$P_{new} \notin SliceConds_{src}(D), P_{old} \Rightarrow P_{new}$	
Postcondition: $SliceConds_{trg}(D) =$	$\mathbf{Dim} = D$
$SliceConds_{src}(D) \setminus \{P_{old}\} \cup \{P_{new}\}$	$\mathbf{SC} \colon P_{old} \to P_{new}$
refocusSliceCond (D, P_1, \cdots, P_s)	refocusSliceCond
Precondition: $P_1, \dots, P_s \in DimPredicates_D$,	
$\{P_1, \cdots, P_s\} \neq SliceConds_{src}(D)$	
Postcondition: $SliceConds_{trg}(D) =$	$\mathbf{Dim} = D$
$\{P_1,\cdots,P_s\}$	$\mathbf{SC}: P_1, \cdots, P_s$
refocusSliceCond(D , $P_{old} \rightarrow P_{new}$)	refocusSliceCond
Precondition: $P_{old}, P_{new} \in DimPredicates_D$,	
$P_{old} \in SliceConds_{src}(D), P_{new} \notin SliceConds_{src}(D)$	
Postcondition: $SliceConds_{trg}(D) =$	$\mathbf{Dim} = D$
$SliceConds_{src}(D) \setminus \{P_{old}\} \cup \{P_{new}\}$	$\mathbf{SC} \colon P_{old} \to P_{new}$

Table 3.5: Operators narrowBMsrCond, broadenBMsrCond, and refocus-BMsrCond

BMsrCond	
Operator Definition	Symbol
$\frac{\texttt{narrowBMsrCond}(+B_1,\cdots,+B_b)}{}$	narrowBMsrCond
Precondition:	
$B_1, \cdots, B_b \in BMsrPredicates_C \text{ with } C =$	12
$CubeSchema_{src}, \{B_1, \cdots, B_b\} \nsubseteq BMsrConds_{src}$	$\mathbf{BC}\colon +B_1,\cdots,+B_b$
Postcondition: $BMsrConds_{trg} =$	$\mathbf{BC}: + D_1, \cdots, + D_b$
$BMsrConds_{src} \cup \{B_1, \cdots, B_b\}$	
$\underline{\text{narrowBMsrCond}(B_{old} \rightarrow B_{new})}$	
Precondition: $B_{new} \in BMsrPredicates_C$	narrowBMsrCond
with $C = CubeSchema_{src}$,	
$B_{old} \in BMsrConds_{src},$	12
$B_{new} \notin BMsrConds_{src}, B_{new} \Rightarrow B_{old}$	$\mathbf{PC} : R \longrightarrow R$
Postcondition: $BMsrConds_{trg} =$	$\mathbf{BC} \colon B_{old} \to B_{new}$
$BMsrConds_{src} \setminus \{B_{old}\} \cup \{B_{new}\}$	
$\underline{\text{broadenBMsrCond}(-B_1,\cdots,-B_b)}$	broadenBMsrCond
Precondition:	
$\{B_1, \cdots, B_b\} \subseteq BMsrConds_{src}$	
Postcondition: $BMsrConds_{trg} =$	$\mathbf{p}_{\mathbf{C}}$, \mathbf{p}
$BMsrConds_{src} \setminus \{B_1, \cdots, B_b\}$	$\mathbf{BC}\colon -B_1,\cdots,-B_b$
$\underline{\text{broadenBMsrCond}(B_{old} \rightarrow B_{new})}$	
Precondition:	broadenBMsrCond
$B_{new} \in BMsrPredicates_C \text{ with } C = CubeSchema_{src},$	
$B_{old} \in BMsrConds_{src},$	12
$B_{new} \notin BMsrConds_{src}, B_{old} \Rightarrow B_{new}$	$\mathbf{BC} \colon B_{old} \to B_{new}$
Postcondition: $BMsrConds_{trg} =$	\mathbf{BC} . $D_{old} \rightarrow D_{new}$
$BMsrConds_{src} \setminus \{B_{old}\} \cup \{B_{new}\}$	
$refocusBMsrCond(B_1, \cdots, B_b)$	refocusBMsrCond
Precondition: $B_1, \dots, B_b \in BMsrPredicates_C$	(12)
with $C = CubeSchema_{src}$	12
$BMsrConds_{src} \neq \{B_1, \cdots, B_b\}$	$\mathbf{BC} \colon B_1, \cdots, B_b$
Postcondition: $BMsrConds_{trg} = \{B_1, \dots, B_b\}$	$BO: D_1, \cdots, D_b$
$\underline{\text{refocusBMsrCond}(B_{old} \rightarrow B_{new})}$	refocusBMsrCond
Precondition:	(12)
$B_{new} \in BMsrPredicates_C \text{ with } C = CubeSchema_{src},$	12
$B_{old} \in BMsrConds_{src}, B_{new} \notin BMsrConds_{src}$	$\mathbf{p}_{\mathbf{C}}$, $\mathbf{p}_{\mathbf{c}}$, $\mathbf{p}_{\mathbf{c}}$
Postcondition: $BMsrConds_{trg} =$	$\mathbf{BC} \colon B_{old} \to B_{new}$
$BMsrConds_{src} \setminus \{B_{old}\} \cup \{B_{new}\}$	

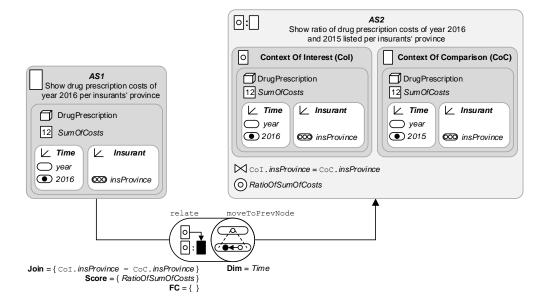


Figure 3.2: Example of a comparative navigation step

3.2.5 Operators Changing Measures

see Table 3.6

3.2.6 Operators Changing Filters

see Table 3.7

3.2.7 Operators Changing Cube Access

see Table 3.8

3.3 Operators Involving Comparison

see Figure 3.2

3.3.1 Operators Introducing Comparison

see Table 3.9

Table 3.6: Operators addMeasure, dropMeasure, refocusMeasure, moveDownToMeasure, and moveUpToMeasure

DownToMeasure, and moveUpToMeasure	
Operator Definition	Symbol
addMeasure(+ M_1 ,,+ M_m)	
Precondition:	addMeasure
$M_1, \dots, M_m \in Msrs_C \text{ with } C = CubeSchema_{src},$	dddileddire
$\{M_1,\cdots,M_m\} \nsubseteq Msrs_{src}$	$\left(\begin{array}{c} \left(\begin{array}{c} 12 \\ 34 \end{array}\right) \right)$
Postcondition:	
$Msrs_{trg} = Msrs_{src} \cup \{M_1, \cdots, M_m\}$	$\mathbf{Msr}\colon +M_1,\cdots,+M_m$
dropMeasure(- M_1 ,,- M_m)	dropMeasure
Precondition:	ur opneusur c
$\{M_1,\cdots,M_m\}\subseteq Msrs_{src}$	
Postcondition:	×
$Msrs_{trg} = Msrs_{src} \setminus \{M_1, \cdots, M_m\}$	$\mathbf{Msr}\colon -M_1,\cdots,-M_m$
$refocusMeasure(M_1, \cdots, M_m)$	
Precondition:	refocusMeasure
$M_1, \dots, M_m \in Msrs_C \text{ with } C = CubeSchema_{src},$	
$\{M_1,\cdots,M_m\} \neq Msrs_{src}$	(12 34)
Postcondition:	
$Msrs_{trg} = \{M_1, \cdots, M_m\}$	$\mathbf{Msr} = M_1, \cdots, M_m$
$refocusMeasure(M_{old} \rightarrow M_{new})$	
Precondition:	refocusMeasure
$M_{new} \in Msrs_C \text{ with } C = CubeSchema_{src},$	
$M_{old} \in Msrs_{src}, M_{new} \notin Msrs_{src}$	(12 34)
Postcondition: $Msrs_{trg} =$	Man. M. M.
$Msrs_{src} \setminus \{M_{old}\} \cup \{M_{new}\}$	$\mathbf{Msr} \colon M_{old} \to M_{new}$
$\underline{\text{moveDownToMeasure}(M_{old} \rightarrow M_{new})}$	
Precondition:	moveDownToMeasure
$M_{new} \in Msrs_C \text{ with } C = CubeSchema_{src},$	12-
$M_{new} \notin Msrs_{src}, M_{new} \twoheadrightarrow M_{old}$	
Postcondition:	$M_{\mathbf{cr}} : M \longrightarrow M$
$Msrs_{trg} = Msrs_{src} \setminus \{M_{old}\} \cup \{M_{new}\}$	$\mathbf{Msr} \colon M_{old} \to M_{new}$
moveUpToMeasure($M_{old} \rightarrow M_{new}$)	
Precondition:	moveUpToMeasure
$M_{new} \in Msrs_C \text{ with } C = CubeSchema_{src},$	124
$M_{new} \notin Msrs_{src}, M_{old} \twoheadrightarrow M_{new}$	34
Postcondition:	Men: M
$Msrs_{trg} = Msrs_{src} \setminus \{M_{old}\} \cup \{M_{new}\}$	$\mathbf{Msr} \colon M_{old} \to M_{new}$

Table 3.7: Operator narrowFilter

Table 3.7: Operator narrowFilt	
Operator Definition	Symbol
$\underline{\text{narrowFilter}(+F_1,\cdots,+F_k)}$	
Precondition:	narrowFilter
$F_1, \dots, F_k \in MsrPredicates_C \text{ with } C =$	
$CubeSchema_{src}, \{F_1, \cdots, F_k\} \nsubseteq FilterConds_{src}$	
Postcondition: $FilterConds_{trg} =$	FC: F + E
$FilterConds_{src} \cup \{F_1, \cdots, F_k\}$	$\mathbf{FC}: +F_1, \cdots, +F_k$
$\underline{\text{narrowFilter}(F_{old} \rightarrow F_{new})}$	
Precondition:	
$F_{new} \in MsrPredicates_C \text{ with } C = CubeSchema_{src},$	narrowFilter
$F_{old} \in FilterConds_{src},$	
$F_{new} \notin FilterConds_{src}, F_{new} \Rightarrow F_{old}$	
Postcondition: $FilterConds_{trg} =$	$\mathbf{FC} \colon F_{old} \to F_{new}$
$FilterConds_{src} \setminus \{F_{old}\} \cup \{F_{new}\}$	$\mathbf{FC} \colon F_{old} \to F_{new}$
$\underline{\text{broadenFilter}(\neg F_1, \cdots, \neg F_k)}$	broadenFilter
Precondition:	
$\{F_1, \cdots, F_k\} \subseteq FilterConds_{src}$	
Postcondition: $FilterConds_{trg} =$	FC E
$FilterConds_{src} \setminus \{F_1, \cdots, F_k\}$	$\mathbf{FC}: -F_1, \cdots, -F_k$
$\underline{\text{broadenFilter}(F_{old} \rightarrow F_{new})}$	
Precondition:	
$F_{new} \in MsrPredicates_C \text{ with } C = CubeSchema_{src},$	broadenFilter
$F_{old} \in FilterConds_{src},$	
$F_{new} \notin FilterConds_{src}, F_{old} \Rightarrow F_{new}$	
Postcondition: $FilterConds_{trg} =$	DC E
$FilterConds_{src} \setminus \{F_{old}\} \cup \{F_{new}\}$	$\mathbf{FC} \colon F_{old} \to F_{new}$
refocusFilter (F_1, \dots, F_k)	
Precondition:	refocusFilter
$F_1, \dots, F_k \in MsrPredicates_C \text{ with } C = CubeSchema_{src}$	
$FilterConds_{src} \neq \{F_1, \cdots, F_k\}$	
Postcondition: $FilterConds_{trg} = \{F_1, \dots, F_k\}$	
	$\mathbf{FC}: F_1, \cdots, F_k$
$\underline{\text{refocusFilter}(F_{old} \rightarrow F_{new})}$	
Precondition:	refocusFilter
$F_{new} \in MsrPredicates_C \text{ with } C = CubeSchema_{src},$	
$F_{old} \in FilterConds_{src}, F_{new} \notin FilterConds_{src}$	
Postcondition: $FilterConds_{trg} =$	PC F
$FilterConds_{src} \setminus \{F_{old}\} \cup \{F_{new}\}$	$\mathbf{FC} \colon F_{old} \to F_{new}$

Table 3.8: Operator drillAcrossToCube

Operator Definition	Symbol
drillAcrossToCube(C,B,M,F)	drillAcrossToCube
Precondition: C is a cube, $B \subseteq BMsrPredicates_C$,	
$M \subseteq Msrs_C, M \neq \emptyset$, and $F \subseteq MsrPredicates_C$	
Postcondition:	
$CubeInstance_{trg} = C, BMsrConds_{trg} = B, and$	$\mathbf{Cube} = C$
if $D \in DimSchemas_{src} \cap DimSchemas_C$,	$\mathbf{BC} = B$
then $DimQual_{src}(D) \in DimQuals_{trg}$,	$\mathbf{Msr} = M$
if $D \in DimSchemas_C \land D \notin DimSchemas_{src}$,	$\mathbf{FC} = F$
then $DiceLvl_{dq} = top_D$, $DiceNode_{dq} = all_D$,	
$SliceConds_{dq} = \emptyset$, and $GranLvl_{dq} = top_D$ with	
$dq = DimQual_{trg}(D)$, and moreover, $Msrs_{trg} = M$	
and $FilterConds_{trg} = F$	

3.3.2 Operators Changing Comparison

see Table 3.10, 3.11, 3.12, 3.13, 3.14

3.3.3 Operators Dropping Comparison

see Table 3.15

3.4 Use of Analysis Situations as Cubes

see Table 3.16 see Figure 3.3, 3.4

Table 3.9: Operators relate and target

Operator Definition	Symbol
relate $(OP(p_1, \dots, p_q), J, S, F^{\circ})$ Precondition: J is a non-empty set of join conditions between src and $src.OP(p_1, \dots, p_q)$, S is a non-empty set of scores over src and $src.OP(p_1, \dots, p_q)$, and F° is a possibly empty set of score filters over src and $src.OP(p_1, \dots, p_q)$ with scores of S . Postcondition: $CoI_{trg} = src$, $CoC_{trg} = src.OP(p_1, \dots, p_q)$, $JoinConds_{trg} = J$, $Scores_{trg} = S$, $ScoreFilters_{trg} = F^{\circ}$	$egin{array}{cccccccccccccccccccccccccccccccccccc$
relate (J, S, F°) Precondition: J is a non-empty set of join conditions between src and src , S is a non-empty set of scores over src and src , and F° is a possibly empty set of score filters over src and src with scores of S . Postcondition: $CoI_{trg} = src$, $CoC_{trg} = src$, $JoinConds_{trg} = J$, $Scores_{trg} = S$, $ScoreFilters_{trg} = F^{\circ}$	$egin{array}{c} \mathbf{relate} \\ \hline egin{array}{c} \mathbf{O} \\ \hline \mathbf{O} \end{bmatrix} \\ \mathbf{Join} = J \\ \mathbf{Score} = S \\ \mathbf{FC} = F^{\circ} \\ \end{array}$
target $(OP(p_1, \dots, p_q), J, S, F^{\circ})$ Precondition: J is a non-empty set of join conditions between src and $src.OP(p_1, \dots, p_q)$, S is a non-empty set of scores over src and $src.OP(p_1, \dots, p_q)$, and F° is a possibly empty set of score filters over src and $src.OP(p_1, \dots, p_q)$ with scores of S . Postcondition: $CoC_{trg} = src$, $CoI_{trg} = src.OP(p_1, \dots, p_q)$, $JoinConds_{trg} = J$, $Scores_{trg} = S$, $ScoreFilters_{trg} = F^{\circ}$	$\mathbf{Join} = J p_1$ $\mathbf{Score} = S \cdots$ $\mathbf{FC} = F^{\circ} p_q$

Table 3.10: Operator rerelate

Operator Definition	Symbol
rerelate($OP(p_1, \cdots, p_q)$, J)	
Precondition: J is a non-empty set of join	rerelate OP
conditions between CoI_{src} and CoC_{src} . $OP(p_1,$	@:D-
$ \cdots,p_q\rangle$	
Postcondition: $CoI_{trg} = CoI_{src}$,	$\mathbf{Join} = J p_1$
$CoC_{trg} = CoC_{src}.OP(p_1, \cdots, p_q),$	
$JoinConds_{trg} = J, Scores_{trg} = Scores_{src},$	p_q
$ScoreFilters_{trg} = ScoreFilters_{src}$	- 1
rerelate($OP(p_1, \cdots, p_q)$, J , S , F°)	
Precondition: J is a non-empty set of join	
conditions between CoI_{src} and CoC_{src} . $OP(p_1,$	1 . OD
\cdots , p_q), S is a non-empty set of scores over	rerelate <i>OP</i>
CoI_{src} and CoC_{src} . $OP(p_1, \dots, p_q)$, and F°	
is a possibly empty set of score filters over CoI_{src}	
and CoC_{src} . $OP(p_1, \dots, p_q)$ with scores of S .	$\mathbf{Join} = J p_1$
Postcondition: $CoI_{trg} = CoI_{src}$,	$\mathbf{Score} = S \cdots$
$CoC_{trg} = CoC_{src}.OP(p_1, \cdots, p_q),$	$\mathbf{FC} = F^{\circ} p_q$
$JoinConds_{trg} = J, Scores_{trg} = S,$	
$ScoreFilters_{trg} = F^{\circ}$	
$rerelate(OP(p_1, \cdots, p_q))$	
Precondition: $JoinConds_{src}$ are also join	
conditions between CoI_{src} and CoC_{src} . $OP(p_1,$	1 . OD
\cdots, p_q), $Scores_{src}$ are also scores over CoI_{src}	rerelate <i>OP</i>
and CoC_{src} . $OP(p_1, \dots, p_q)$, and $ScoreFilters_{src}$	
are also scores over filters over CoI_{src} and	
CoC_{src} . $OP(p_1, \dots, p_q)$ with scores $Scores_{src}$.	p_1
Postcondition: $CoI_{trg} = CoI_{src}$,	•••
$CoC_{trg} = CoC_{src}.OP(p_1, \cdots, p_q),$	p_q
$JoinConds_{trg} = JoinConds_{src}, Scores_{trg} =$	
$Scores_{src}, ScoreFilters_{trg} = ScoreFilters_{src}$	

Table 3.11: Operator retarget

Operator Definition	Symbol
retarget($OP(p_1, \dots, p_q), J$)	O.D.
Precondition: J is a non-empty set of join	retarget <i>OP</i>
conditions between CoI_{src} . $OP(p_1, \dots, p_q)$	
and CoC_{src}	
Postcondition:	$\mathbf{Join} = J p_1$
$CoI_{trg} = CoI_{src} \cdot OP(p_1, \cdots, p_q), CoC_{trg} =$	• • •
CoC_{src} , $JoinConds_{trg} = J$, $Scores_{trg} = Scores_{src}$,	p_q
$ScoreFilters_{trg} = ScoreFilters_{src}$	-
$retarget(\mathit{OP}(p_1, \cdots, p_q), J, S, F^\circ)$	
Precondition: J is a non-empty set of join	
conditions between CoI_{src} . $OP(p_1, \dots, p_q)$	
and CoC_{src} , S is a non-empty set of scores over	2-
CoI_{src} . $OP(p_1, \dots, p_q)$ and CoC_{src} , and F°	retarget <i>OP</i>
is a possibly empty set of score filters over	
$CoI_{src}.OP(p_1, \cdots, p_q)$ and CoC_{src} with	
scores of S .	$\mathbf{Join} = J p_1$
Postcondition:	$\mathbf{Score} = S \cdots$
$CoI_{trq} = CoI_{src}.OP(p_1, \cdots, p_q),$	$\mathbf{FC} = F^{\circ} p_q$
$CoC_{trq} = CoC_{src}, JoinConds_{trq} = J,$	- 1
$Scores_{trq} = S, ScoreFilters_{trq} = F^{\circ}$	
$retarget(OP(p_1, \dots, p_q))$	
Precondition: $JoinConds_{src}$ are also join	
conditions between CoI_{src} . $OP(p_1, \dots, p_q)$	
and CoC_{src} , $Scores_{src}$ are also scores over	
and CoI_{src} . $OP(p_1, \dots, p_q)$ and CoC_{src} , and	retarget <i>OP</i>
$ScoreFilters_{src}$ are also scores over filters over	
CoI_{src} . $OP(p_1, \dots, p_q)$ and CoC_{src} with	
scores $Scores_{src}$.	p_1
Postcondition: $CoI_{trg} =$	
CoI_{src} . $OP(p_1, \dots, p_q)$, $CoC_{trg} = CoC_{src}$,	p_q
$JoinConds_{trg} = JoinConds_{src}, Scores_{trg} =$	_
$Scores_{src}, ScoreFilters_{trg} = ScoreFilters_{src}$	

Table 3.12: Operator correlate

Operator Definition	Symbol
$correlate(OP(p_1, \cdots, p_q), J)$	
Precondition: J is a non-empty set of join	correlate OP
conditions between CoI_{src} . $OP(p_1, \dots, p_q)$ and	
CoC_{src} . $OP(p_1, \cdots, p_q)$	
Postcondition: $CoI_{trg} = CoI_{src} \cdot OP(p_1, \cdots, p_n)$	$\mathbf{Join} = J p_1$
p_q , $CoC_{trg} = CoC_{src} \cdot OP(p_1, \cdots, p_q)$,	
$JoinConds_{trg} = J, Scores_{trg} = Scores_{src},$	p_q
$ScoreFilters_{trg} = ScoreFilters_{src}$	
correlate($OP(p_1, \cdots, p_q), J, S, F^{\circ}$)	
Precondition: J is a non-empty set of join	
conditions between CoI_{src} . $OP(p_1, \dots, p_q)$ and	
CoC_{src} . $OP(p_1, \dots, p_q)$, S is a non-empty set of)
scores over CoI_{src} . $OP(p_1, \dots, p_q)$ and	correlate <i>OP</i>
CoC_{src} . $OP(p_1, \dots, p_q)$, and F° is a possibly	
empty set of score filters over CoI_{src} . $OP(p_1, \dots, p_n)$	
p_q) and CoC_{src} . $OP(p_1, \dots, p_q)$ with scores of S .	$\mathbf{Join} = J p_1$
Postcondition: $CoI_{trg} = CoI_{src} \cdot OP(p_1, \cdots, p_{src})$	$\mathbf{Score} = S \cdots$
p_q , $CoC_{trg} = CoC_{src} \cdot OP(p_1, \dots, p_q)$,	$\mathbf{FC} = F^{\circ} p_q$
$JoinConds_{trg} = J, Scores_{trg} = S,$	
$ScoreFilters_{trg} = F^{\circ}$	
$correlate(OP(p_1, \dots, p_q))$	
Precondition: $JoinConds_{src}$ are also join	
conditions between CoI_{src} . $OP(p_1, \dots, p_q)$ and	
CoC_{src} . $OP(p_1, \dots, p_q)$, $Scores_{src}$ are also scores	correlate <i>OP</i>
over CoI_{src} . $OP(p_1, \dots, p_q)$ and CoC_{src} . $OP(p_1, \dots, p_q)$	correlate <i>OP</i>
\cdots , p_q), and $ScoreFilters_{src}$ are also scores over	
filters over CoI_{src} . $OP(p_1, \dots, p_q)$ and	
CoC_{src} . $OP(p_1, \dots, p_q)$ with scores $Scores_{src}$.	p_1
Postcondition: $CoI_{trg} = CoI_{src} \cdot OP(p_1, \cdots, p_{src})$	•••
p_q), $CoC_{trg} = CoC_{src} \cdot OP(p_1, \dots, p_q)$,	p_q
$JoinConds_{trg} = JoinConds_{src}, Scores_{trg} =$	
$Scores_{src}, ScoreFilters_{trg} = ScoreFilters_{src}$	

 ${\bf Table~3.13:~Operators~narrowFilter,~broadenFilter,~and~refocusFilter}$

for score filters	,
Operator Definition	Symbol
narrowFilter(+ F_1° , \cdots ,+ F_w°)	
Precondition: $F_1^{\circ}, \dots, F_w^{\circ}$ are score filters	narrowFilter
over CoI_{src} and CoC_{src} with scores in $Scores_{src}$,	
and $\{F_1^{\circ}, \cdots, F_w^{\circ}\} \nsubseteq ScoreFilters_{src}$.	
Postcondition:	$\mathbf{FC} \colon +F_1^{\circ}, \cdots, +F_w^{\circ}$
$ScoreFilters_{trg} = ScoreFilters_{src} \cup \{F_1^{\circ}, \cdots, F_w^{\circ}\}$	1, 1, ,, w
$ ext{narrowFilter}(F_{old}^{\circ} ext{ -> } F_{new}^{\circ})$	
Precondition: F_{new}° is a score filter over CoI_{src}	narrowFilter
and CoC_{src} with scores in $Scores_{src}$,	
$F_{old}^{\circ} \in ScoreFilters_{src}, F_{new}^{\circ} \notin ScoreFilters_{src},$	
and $F_{new}^{\circ} \Rightarrow F_{old}^{\circ}$.	$\mathbf{FC} \cdot F^{\circ} \qquad F^{\circ}$
Postcondition:	$\mathbf{FC} \colon F_{old}^{\circ} \to F_{new}^{\circ}$
$ScoreFilters_{trg} = ScoreFilters_{src} \setminus \{F_{old}^{\circ}\} \cup \{F_{new}^{\circ}\}$	
$\underline{\text{broadenFilter}(\neg F_1^{\circ}, \cdots, \neg F_w^{\circ})}$	broadenFilter
Precondition: $\{F_1^{\circ}, \cdots, F_w^{\circ}\} \subseteq ScoreFilters_{src}$.	
Postcondition:	
$ScoreFilters_{trg} = ScoreFilters_{src} \setminus \{F_1^{\circ}, \cdots, F_w^{\circ}\}$	$\mathbf{FC}\colon -F_1^{\circ}, \cdots, -F_w^{\circ}$
broadenFilter($F_{old}^{\circ} \rightarrow F_{new}^{\circ}$)	
Precondition: F_{new}° is a score filter over CoI_{src}	
and CoC_{src} with scores in $Scores_{src}$, and	broadenFilter
$F_{old}^{\circ} \in ScoreFilters_{src}, F_{new}^{\circ} \notin ScoreFilters_{src},$	
and $F_{old}^{\circ} \Rightarrow F_{new}^{\circ}$.	
Postcondition:	$\mathbf{FC} \colon F_{old}^{\circ} \to F_{new}^{\circ}$
$ScoreFilters_{trg} = ScoreFilters_{src} \setminus \{F_{old}^{\circ}\} \cup \{F_{new}^{\circ}\}$	ota new
refocusFilter($F_1^{\circ},\cdots,F_w^{\circ}$)	
Precondition: $F_1^{\circ}, \dots, F_w^{\circ}$ are score filters	refocusFilter
over CoI_{src} and CoC_{src} with scores in $Scores_{src}$,	
and $\{F_1^{\circ}, \cdots, F_w^{\circ}\} \neq ScoreFilters_{src}$.	
Postcondition:	$\mathbf{FC} \colon F_1^{\circ}, \cdots, F_w^{\circ}$
$ScoreFilters_{trg} = \{F_1^{\circ}, \cdots, F_w^{\circ}\}$ $refocusFilter(F_{old}^{\circ} \rightarrow F_{new}^{\circ})$	
$\frac{\text{refocusFilter}(F_{old}^{\circ} \rightarrow F_{new}^{\circ})}{}$	
Precondition: F_{new}° is a score filter over CoI_{src}	refocusFilter
and CoC_{src} with scores in $Scores_{src}$,	
$F_{old}^{\circ} \in ScoreFilters_{src}$, and $F_{new}^{\circ} \notin ScoreFilters_{src}$.	
Postcondition:	$\mathbf{FC} \colon F_{old}^{\circ} \to F_{new}^{\circ}$
$ScoreFilters_{trg} = ScoreFilters_{src} \setminus \{F_{old}^{\circ}\} \cup \{F_{new}^{\circ}\}$	

Table 3.14: Operator rejoin

Operator Definition	Symbol
rejoin(J)	rejoin
Precondition: J is a set of join conditions between	Tejoin
CoI_{src} and CoC_{src} ,	
Postcondition:	Tain I
$JoinConds_{trg} = J, Scores_{trg} = Scores_{src},$	$\mathbf{Join} = J$
$ScoreFilters_{trg} = ScoreFilters_{src}$	
$rejoin(J,S,F^{\circ})$	
Precondition: J is a set of join conditions between	rejoin
CoI_{src} and CoC_{src} , S is a set of scores over CoI_{src} and	
CoC_{src} , and F are score filters over CoI_{src} and CoC_{src}	
with scores in S .	$\mathbf{Join} = J$
Postcondition:	$\mathbf{Score} = S$
$JoinConds_{trg} = J, Scores_{trg} = S, ScoreFilters_{trg} = F$	$\mathbf{FC} = F^{\circ}$

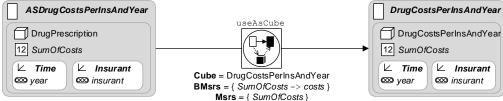
Table 3.15: Operators unrelate and untarget

Operator Definition	Symbol
unrelate()	unrelate
Precondition:	(FOI:T)
No additional preconditions	
Postcondition: $trg = CoC_{src}$	
untarget()	untarget
Precondition:	
No additional preconditions	
Postcondition: $trg = CoI_{src}$	

Table 3.16: Operator useAsCube

Operator Definition Symbol useAsCube(C, \mathcal{B} , M) **Precondition:** src is a non-comparative analysis useAsCube situation, C is a cube schema derived from srcsuch that $BMsrs_C = \mathcal{B}(Msrs_{src})$ where \mathcal{B} is a base measure renaming from src to C, and M are Cube = Cmeasures applicable to $BMsrs_C$. **Postcondition:** *trg* is a non-comparative analysis, $\mathbf{BMsrs} = \mathcal{B}$ situation, $CubeSchema_{trg} = C$, $Msrs_{trg} = M$, for Msrs = Mall $D \in CubeSchema_C$, $DiceLvl_{trq}(D) = top_D$, $DiceNode_{trq}(D) = all_D, SliceConds_{trq}(D) = \emptyset,$ $GranLvl_{trq}(D) = base_D$, and $FilterConds_{trq} = \emptyset$. $\underline{\mathtt{useAsCube}(C},\ \mathcal{B},\ \mathcal{D},\ \mathcal{L},\ \mathcal{A},\ M)$ useAsCube **Precondition:** src is a comparative analysis situation, C is a cube schema derived from srcsuch that $BMsrs_C = \mathcal{B}(Msrs_{CoI_{src}} \dot{\cup} Msrs_{CoI_{src}} \dot{\cup}$ $Scores_{src}$) where \mathcal{B} is a base measure renaming Cube = C $\mathbf{BMsrs} = \mathcal{M}$ from src to C, and M are measures applicable $\mathbf{Dims} = \mathcal{D}$ to $BMsrs_C$. Moreover, dimension schemas of $\mathbf{Lvls} = \mathcal{L}$ $DimSchemas_C$ are renamed by \mathcal{D} , the levels are $\mathbf{Attrs} = \mathcal{A}$ renamed by \mathcal{L} , and descriptive attributes by \mathcal{A} . **Postcondition:** *trg* is a non-comparative analysis, Msrs = Msituation, $CubeSchema_{trq} = C$, $Msrs_{trq} = M$, for all $D \in CubeSchema_C$, $DiceLvl_{trq}(D) = top_D$, $DiceNode_{trq}(D) = all_D, SliceConds_{trg}(D) = \emptyset,$ $GranLvl_{trq}(D) = base_D$, and $FilterConds_{trq} = \emptyset$.





SQL translation:

create view DrugCostsPerInsAndYear as year, insDistrict, SUM(costs) as costs select

DrugPrescription natural join Time natural join Insurant

group by

eDFM of the derived cube schema:

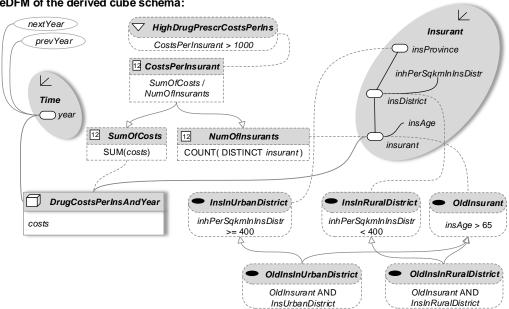
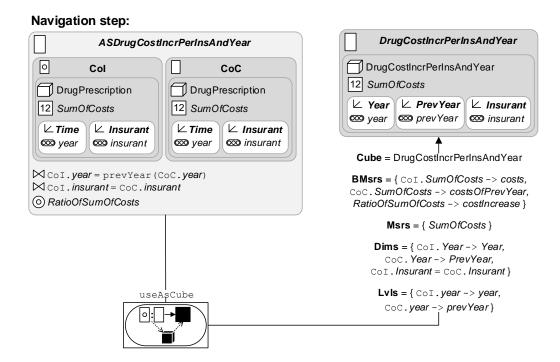


Figure 3.3: Operator useAsCube (non-comparative analysis situation)



SQL translation:

create view DrugCostIncrPerInsAndYear as

select Col.year as year, Col.insDistrict as insDistrict, Col.SumOfCosts as costs, CoC.year as prevYear, CoC.SumOfCosts as costsOfPrevYear, Col.SumOfCosts / CoC.SumOfCosts as costIncrease

from (select year, insDistrict, SUM(costs) as SumOfCosts from DrugPrescription natural join Time natural join Insurant group by year, insDistrict) Col

inner join (select year, insDistrict, SUM(costs) as SumOfCosts

from DrugPrescription natural join Time natural join Insurant

group by year, insDistrict) CoC

on Col.year = CoC.year + 1

Figure 3.4: Operator useAsCube (comparative analysis situation)

Chapter 4

Extension to Schema Level

4.1 Analysis Situation Schema

see Figure 4.1, 4.2

4.2 Navigation Step Schema

see Figure 4.3, 4.4, 4.5

4.3 Navigation guards

see Table 4.1, 4.2, 4.4, 4.5 see Figure 4.6, 4.7

Table 4.1: Navigation guard expressions concerning granularity levels (as denotes a non-comparative and cas a comparative analysis situation)

Operator definition as.granLevel(D): If $D \in DimSchemas_{as}$, then as granLevel(D) = $GranLvl_{as}(D)$. $cas. {\tt granLevelOfCoI}(\overline{D})$: If $D \in DimSchemas_{CoI_{cas}}$, then $cas.granLevelOfCoI(D) = GranLvl_{CoI_{cas}}(D)$. cas.granLevelOfCoC(D): If $D \in DimSchemas_{CoC_{cas}}$, then $cas.granLevelOfCoC(D) = GranLvl_{CoC_{cas}}(D)$. as.hasSuperGranLevel(D): If $D \in DimSchemas_{as}$, then as.hasSuperGranLevel(D) is true iff $(\exists G \in Lvls_D)(GranLvl_{as}(D) \to G)$ cas.hasSuperGranLevelInCoI(D): If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasSuperGranLevelInCoI(D) is true iff $(\exists G \in Lvls_D)(GranLvl_{CoI_{cas}}(D) \to G)$ $cas. {\tt hasSuperGranLevelInCoC}(D):$ If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasSuperGranLevelInCoC(D) is true iff $(\exists G \in Lvls_D)(GranLvl_{CoC_{cas}}(D) \to G)$ as.hasSubGranLevel(D): If $D \in DimSchemas_{as}$, then as.hasSubGranLevel(D) is true iff $(\exists G \in Lvls_D)(G \to GranLvl_{as}(D))$ cas.hasSubGranLevelInCoI(D): If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasSubGranLevelInCoI(D) is true iff $(\exists G \in Lvls_D)(G \to GranLvl_{CoI_{cas}}(D))$ cas.hasSubGranLevelInCoC(D):

If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasSubGranLevelInCoC(D) is true iff

 $(\exists G \in Lvls_D)(G \to GranLvl_{CoC_{cas}}(D))$

Table 4.2: Navigation guard expressions concerning dice levels (as denotes a non-comparative and cas a comparative analysis situation)

Operator definition

as.diceLevel(D):

If $D \in DimSchemas_{as}$, then as.diceLevel(D) = $DiceLvl_{as}(D)$.

$\underline{\mathit{cas}}.\mathtt{diceLevelOfCoI}(\underline{D})$:

If $D \in DimSchemas_{CoI_{cas}}$, then $cas.diceLevelOfCoI(D) = DiceLvl_{CoI_{cas}}(D)$.

cas.diceLevelOfCoC(D):

If $D \in DimSchemas_{CoC_{cas}}$, then $cas.diceLevelOfCoC(D) = DiceLvl_{CoC_{cas}}(D)$.

as.hasSuperDiceLevel(D):

If $D \in DimSchemas_{as}$, then as.hasSuperDiceLevel(D) is true iff

 $(\exists L \in Lvls_D)(DiceLvl_{as}(D) \to L)$

cas.hasSuperDiceLevelInCoI(D):

If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasSuperDiceLevelInCoI(D) is true iff

 $(\exists L \in Lvls_D)(DiceLvl_{CoI_{cas}}(D) \to L)$

cas.hasSuperDiceLevelInCoC(D):

If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasSuperDiceLevelInCoC(D) is true iff

 $(\exists L \in Lvls_D)(DiceLvl_{CoC_{cas}}(D) \to L)$

as.hasSubDiceLevel(D):

If $D \in DimSchemas_{as}$, then as.hasSubDiceLevel(D) is true iff

 $(\exists L \in Lvls_D)(L \to DiceLvl_{as}(D))$

cas.hasSubDiceLevelInCoI(D):

If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasSubDiceLevelInCoI(D) is true iff

 $(\exists L \in Lvls_D)(L \to DiceLvl_{CoI_{cas}}(D))$

cas.hasSubDiceLevelInCoC(D):

If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasSubDiceLevelInCoC(D) is true iff

 $(\exists L \in Lvls_D)(L \to DiceLvl_{CoC_{cas}}(D))$

Table 4.3: Navigation guard expressions concerning dice nodes (as denotes a non-comparative and cas a comparative analysis situation)

Operator definition as.diceNode(D): If $D \in DimSchemas_{as}$, then $as.diceNode(D) = DiceNode_{as}(D)$. cas.diceNodeOfCoI(D): If $D \in DimSchemas_{CoI_{cas}}$, then $cas.diceNodeOfCoI(D) = DiceNode_{CoI_{cas}}(D)$. cas.diceNodeOfCoC(D): If $D \in DimSchemas_{CoC_{cas}}$, then $cas.diceNodeOfCoC(D) = DiceNode_{CoC_{cas}}(D)$. as.hasSuperDiceNode(D): If $D \in DimSchemas_{as}$, then as.hasSuperDiceNode(D) is true iff $(\exists N \in Nodes_{as}(D))(DiceNode_{as}(D) \rightarrow N)$ $cas.\mathtt{hasSuperDiceNodeInCoI}(D):$ If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasSuperDiceNodeInCoI(D) is true iff $(\exists N \in Nodes_{CoI_{as}}(D))(DiceNode_{CoI_{cas}}(D) \rightarrow N)$ cas.hasSuperDiceNodeInCoC(D): If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasSuperDiceNodeInCoC(D) is true iff $(\exists N \in Nodes_{CoC_{as}}(D))(DiceNode_{CoC_{cas}}(D) \rightarrow N)$ as.hasSubDiceNode(D): If $D \in DimSchemas_{as}$, then as.hasSubDiceNode(D) is true iff $(\exists N \in Nodes_{as}(D))(N \rightarrow DiceNode_{as}(D))$ cas.hasSubDiceNodeInCoI(D): If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasSubDiceNodeInCoI(D) is true iff $(\exists N \in Nodes_{CoI_{as}}(D))(N \rightarrow DiceNode_{CoI_{cas}}(D))$ cas.hasSubDiceNodeInCoC(D): If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasSubDiceNodeInCoC(D) is true iff $(\exists N \in Nodes_{CoC_{as}}(D))(N \rightarrow DiceNode_{CoC_{cas}}(D))$

Table 4.4: Navigation guard expressions examining next and previous dice node (as denotes a non-comparative and cas a comparative analysis situation)

Operator definition

as.hasNextDiceNode(D):

If $D \in DimSchemas_{as}$, then as.hasNextDiceNode(D) is true iff $(\exists \hat{N}, N' \in Nodes_{as}(D))(\hat{N} \neq N' \wedge DiceNode_{as}(D) \rightarrow \hat{N} \wedge N' \rightarrow \hat{N} \wedge next(DiceNode_{as}(D)) = N')$

cas.hasNextDiceNodeInCoI(D):

If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasNextDiceNodeInCoI(D) is true iff $(\exists \hat{N}, N' \in Nodes_{CoI_{cas}}(D))(\hat{N} \neq N' \wedge DiceNode_{CoI_{cas}}(D)) \rightarrow \hat{N} \wedge N' \rightarrow \hat{N} \wedge next(DiceNode_{CoI_{cas}}(D)) = N')$

cas.hasNextDiceNodeInCoC(D):

If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasNextDiceNodeInCoC(D) is true iff $(\exists \hat{N}, N' \in Nodes_{CoC_{cas}}(D))(\hat{N} \neq N' \wedge DiceNode_{CoC_{cas}}(D) \rightarrow \hat{N} \wedge N' \rightarrow \hat{N} \wedge next(DiceNode_{CoC_{cas}}(D)) = N')$

as.hasPrevDiceNode(D):

If $D \in DimSchemas_{as}$, then as.hasPrevDiceNode(D) is true iff $(\exists \hat{N}, N' \in Nodes_{as}(D))(\hat{N} \neq N' \wedge DiceNode_{as}(D) \rightarrow \hat{N} \wedge N' \rightarrow \hat{N} \wedge prev(DiceNode_{as}(D)) = N')$

cas.hasPrevDiceNodeInCoI(D):

If $D \in DimSchemas_{CoI_{cas}}$, then cas.hasPrevDiceNodeInCoI(D) is true iff $(\exists \hat{N}, N' \in Nodes_{CoI_{cas}}(D))(\hat{N} \neq N' \wedge DiceNode_{CoI_{cas}}(D) \rightarrow \hat{N} \wedge N' \rightarrow \hat{N} \wedge prev(DiceNode_{CoI_{cas}}(D)) = N')$

$cas.\mathtt{hasPrevDiceNodeInCoC}(D)$:

If $D \in DimSchemas_{CoC_{cas}}$, then cas.hasPrevDiceNodeInCoC(D) is true iff $(\exists \hat{N}, N' \in Nodes_{CoC_{cas}}(D))(\hat{N} \neq N' \wedge DiceNode_{CoC_{cas}}(D) \rightarrow \hat{N} \wedge N' \rightarrow \hat{N} \wedge prev(DiceNode_{CoC_{cas}}(D)) = N')$

Table 4.5: Navigation guard expressions examining slice conditions, filter conditions, score filter conditions, and whether a result set is empty (as denotes a non-comparative and cas a comparative analysis situation)

Operator definition

as.containsSliceCond(D, P):

If $D \in DimSchemas_{as}$ and $P \in DimPredicates_D$, then

as.containsSliceCond(D,P) is true iff $(\exists P' \in SliceConds_{as})(P \Rightarrow P')$

cas.containsSliceCondInCoI(D, P):

If $D \in DimSchemas_{CoI_{cas}}$ and $P \in DimPredicates_D$, then cas.contains-SliceCondInCoI(D, P) is true iff $(\exists P' \in SliceConds_{CoI_{cas}})(P \Rightarrow P')$

cas.containsSliceCondInCoC(D,P):

If $D \in DimSchemas_{CoC_{cas}}$ and $P \in DimPredicates_D$, then cas.contains-SliceCondInCoC(D,P) is true iff $(\exists P' \in SliceConds_{CoC_{cas}})(P \Rightarrow P')$

as.containsBMsrCond(F):

If $B \in BMsrPredicates_{as}$, then

as.containsBMsrCond(B) is true iff $(\exists B' \in BMsrConds_{as})(B \Rightarrow B')$

$cas.\mathtt{containsBMsrCondInCoI}(B):$

If $B \in BMsrPredicates_{CoI_{cas}}$, then cas.containsBMsrCondInCoI(F) is true iff $(\exists B' \in BMsrConds_{CoI_{cas}})(B \Rightarrow B')$

$cas.\mathtt{containsBMsrCondInCoC}(B):$

If $B \in \mathit{BMsrPredicates}_{\mathit{CoC}_{\mathit{cas}}},$ then $\mathit{cas}.\mathtt{containsBMsrCondInCoC}(F)$

is true iff $(\exists B' \in BMsrConds_{CoC_{cas}})(B \Rightarrow B')$

as.containsFilterCond(F):

If $F \in MsrPredicates_{as}$, then

as.containsFilterCond(F) is true iff $(\exists F' \in FilterConds_{as})(F \Rightarrow F')$

cas.containsFilterCondInCoI(F):

If $F \in \mathit{MsrPredicates}_{\mathit{CoI}_{\mathit{cas}}}$, then $\mathit{cas}.\mathtt{containsFilterCondInCoI}(F)$

is true iff $(\exists F' \in FilterConds_{CoI_{cas}})(F \Rightarrow F')$

$\underline{cas}.\mathtt{contains}$ FilterCondInCoC(F):

If $F \in \mathit{MsrPredicates}_{\mathit{CoC}_{\mathit{cas}}},$ then $\mathit{cas}.\mathtt{containsFilterCondInCoC}(F)$

is true iff $(\exists F' \in FilterConds_{CoC_{cas}})(F \Rightarrow F')$

cas.containsScoreFilterCond(F):

If F is a score filter over CoI_{cas} and CoC_{cas} , then

cas.containsScoreFilterCond(F) is true iff $(\exists F' \in ScoreFilters_{cas})(F \Rightarrow F')$

$as.\mathtt{hasEmptyResult()}$ or $cas.\mathtt{hasEmptyResult()}$:

 $as.\mathtt{hasEmptyResult()}$ ($cas.\mathtt{hasEmptyResult()}$) is true iff as (cas) has an empty result set.

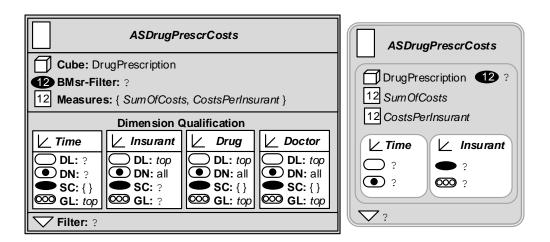


Figure 4.1: Non-comparative analysis situation with unbound variables

O: CompDrug	PrescrCosts			
Context of Interest (Col)	Context of Comparison (CoC)			
Cube: DrugPrescription BMsr-Filter: ? Measures: { SumOfCosts, CostsPerInsurant }	☐ Cube: DrugPrescription ② BMsr-Filter: ? 12 Measures: { SumOfCosts, CostsPerInsurant }			
Dimension Qualification	Dimension Qualification			
SGL: top SGL: ? SGL: top SGL:				
✓ Join Condition: { CoI. insProvince = CoC. insProvince, CoI. top _{Time} = CoC. top _{Time} , CoI. top _{Drug} = CoC. top _{Drug} , CoI. top _{Doctor} = CoC. top _{Doctor} } O Scores: { RatioOfSumOfCosts, RatioOfCostsPerInsurant}				
Score Filter: ?				
○: CompDrugPrescrCosts				
Context Of Interest (Col) DrugPrescription 12 SumOfCosts 12 CostsPerInsurant L Time 2 : 3 ? 2 ?	Context Of Comparison (CoC) DrugPrescription 12 SumOfCosts 12 CostsPerInsurant L Time ? Property of the prop			
CoI.insProvince = CoC.insProvince	surant			

Figure 4.2: Comparative analysis situation with unbound variables

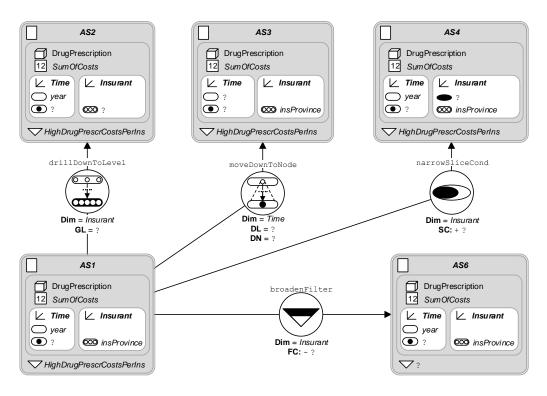


Figure 4.3: Navigation steps with unbound variables

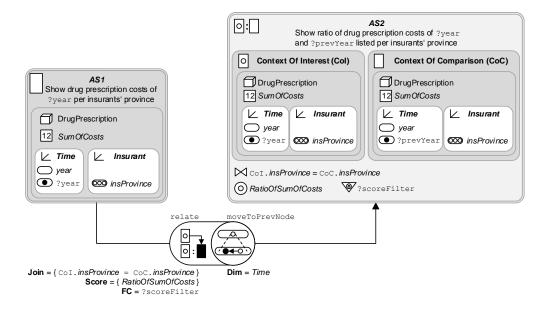


Figure 4.4: Navigation steps with unbound variables involving a comparative analysis situation

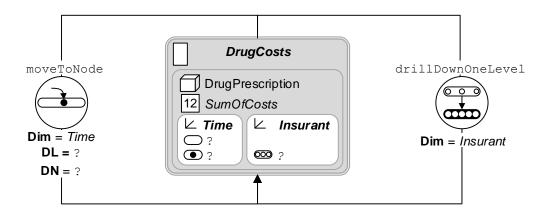


Figure 4.5: Navigation step schemas with equal source and targets

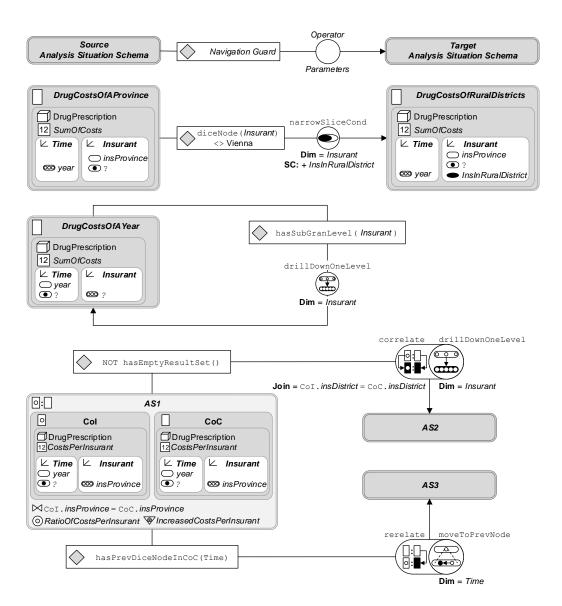


Figure 4.6: Navigation guards

Navigation pattern for refining granularity level of Insurant AS1 AS2 hasSubGranLevel(Insurant) drillDownOneLevel DrugPrescription DrugPrescription (**P**) drillDownOneLevel 12 SumOfCosts 12 SumOfCosts ∠ Time ∠ Time ∠ Insurant ? •? ? Dim = Insurant **∞**? Dim = Insurant Navigation pattern for iterating over all years AS1 AS2 hasNextDiceNode(Time) moveDownToFirstNode DrugPrescription 12 SumOfCosts DrugPrescription 12 SumOfCosts moveToNextNode ∠ Insurant ∠ Insurant ∠ Time ? •? Dim = Time (ÓD) **∞**? **∞**? Dim = Time

Figure 4.7: Navigation patterns

Chapter 5

Business Intelligence (BI) Analysis Graphs

5.1 Definition of BI Analysis Graphs

see Figure 5.1

5.2 BI Analysis Graph Schemas

see Figure 5.2

5.3 Instances of Analysis Graph Schemas

see Figure 5.3

5.4 Composite analysis situation

see Figure 5.4, 5.5, 5.6

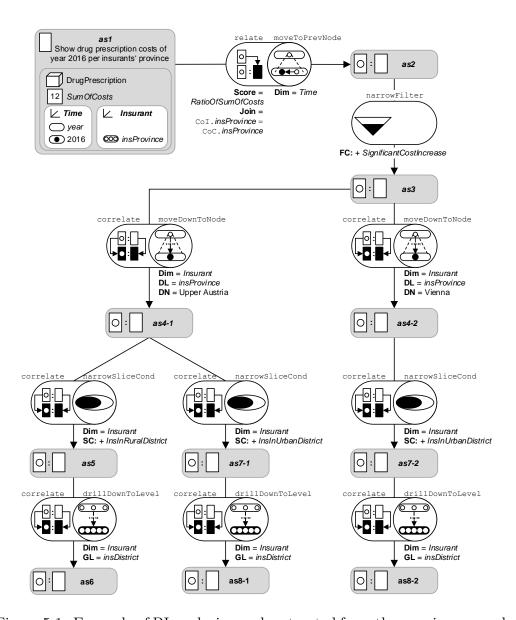


Figure 5.1: Example of BI analysis graph extracted from the running example

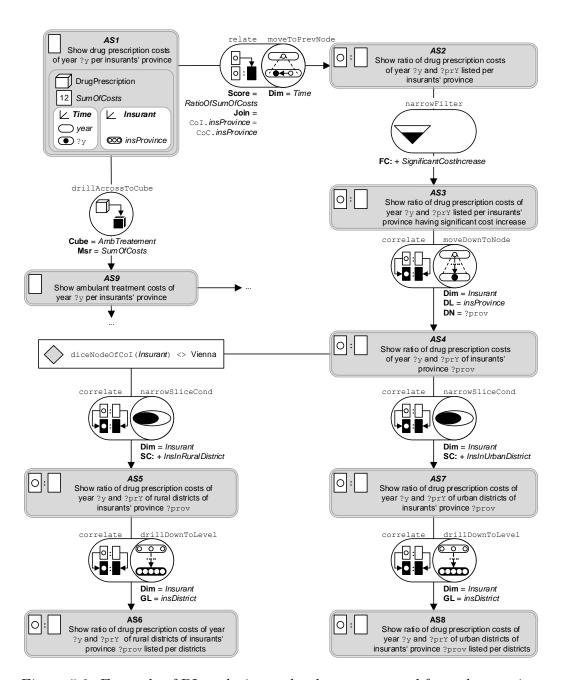


Figure 5.2: Example of BI analysis graph schema extracted from the running example

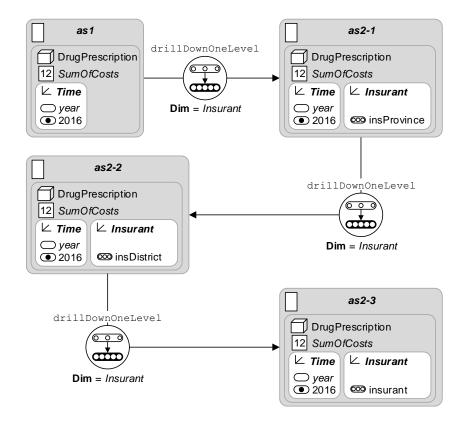


Figure 5.3: Analysis graph instance of analysis graph schema of the first navigation pattern of Figure 4.7

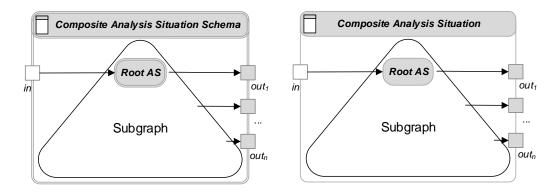


Figure 5.4: Composite analysis situation (schema)

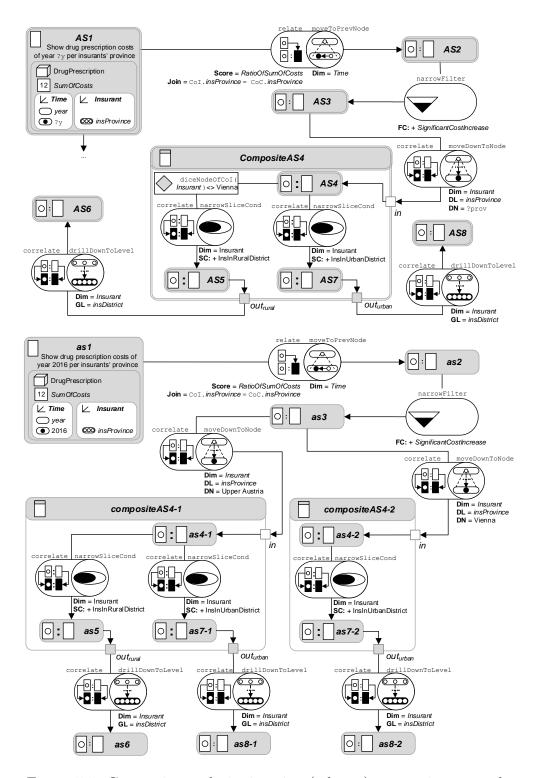


Figure 5.5: Composite analysis situation (schema) — running example

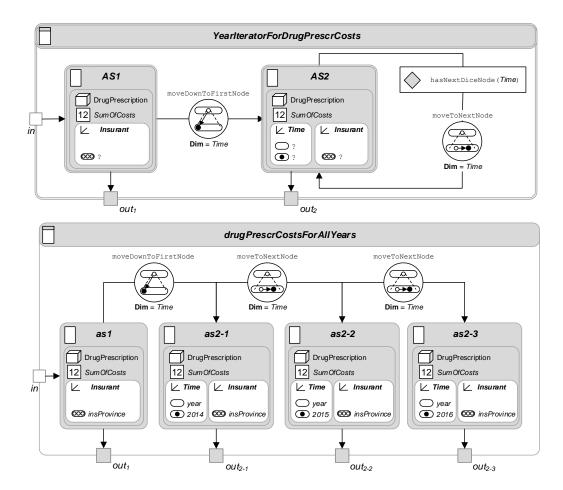


Figure 5.6: Composite analysis situation (schema) example (navigation pattern)

List of Figures

1.1	Health care use case – overview of an analysis process	2
1.2	eDFM for Drug Prescriptions	3
1.3	eDFM for Ambulant Treatment and Hospitalization	4
1.4	Cube Instance	5
2.1	Graphical Template of a Non-comparative Analysis Situation .	8
2.2	Example of a Graphical Representation of a Non-comparative	
	Analysis Situation	8
2.3	Example of an Alternative Graphical Representation of a Non-	
	comparative Analysis Situation	9
2.4	SQL Translation	10
2.5	Example of an SQL Translation	10
2.6	Graphical Template of a Comparative Analysis Situation	11
2.7	Example of a Graphical Representation of a Comparative Ana-	
	lysis Situation	12
2.8	Example of an Alternative Graphical Representation of a Com-	
	parative Analysis Situation	13
2.9	SQL Translation of a Comparative Analysis Situation	14
2.10	Example of an SQL Translation of a Comparative Analysis	
	Situation	15
3.1	Navigation operator examples	18
3.2	Example of a comparative navigation step	24
3.3	Operator ${\tt useAsCube}$ (non-comparative analysis situation)	35
3.4	Operator useAsCube (comparative analysis situation)	36

4.1	Non-comparative analysis situation with unbound variables	43
4.2	Comparative analysis situation with unbound variables	44
4.3	Navigation steps with unbound variables	45
4.4	Navigation steps with unbound variables involving a compar-	
	ative analysis situation	45
4.5	Navigation step schemas with equal source and targets	46
4.6	Navigation guards	47
4.7	Navigation patterns	48
5.1	Example of BI analysis graph extracted from the running ex-	
	ample	50
5.2	Example of BI analysis graph schema extracted from the run-	
	ning example	51
5.3	Analysis graph instance of analysis graph schema of the first	
	navigation pattern of Figure 4.7	52
5.4	Composite analysis situation (schema)	52
5.5	Composite analysis situation (schema) — running example $$	53
5.6	Composite analysis situation (schema) example (navigation	
	pattern)	54

List of Tables

3.1	Operators drillDownOneLevel, drillDownToLevel, rollUp-	
	OneLevel, and rollUpToLevel	19
3.2	$Operators {\tt moveDownToNode}, {\tt moveUpToNode}, {\tt moveAsideToNode}, $	
	and moveToNode	20
3.3	$Operators {\tt moveDownToFirstNode}, {\tt moveDownToLastNode}, {\tt move-lownToLastNode}, {\tt move-$	
	ToNextNode, and moveToPrevNode	21
3.4	Operators narrowSliceCond, broadenSliceCond, and refo-	
	cusSliceCond	22
3.5	${\bf Operators\ narrowBMsrCond}, {\bf broadenBMsrCond}, {\bf and\ refocus-}$	
	BMsrCond	23
3.6	$Operators \ \mathtt{addMeasure}, \ \mathtt{dropMeasure}, \ \mathtt{refocusMeasure}, \ \mathtt{move-}$	
	${\tt DownToMeasure}, \ {\rm and} \ {\tt moveUpToMeasure} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	25
3.7	Operator narrowFilter	26
3.8	Operator drillAcrossToCube	27
3.9	Operators relate and target	28
3.10	Operator rerelate	29
3.11	Operator retarget	30
3.12	Operator correlate	31
3.13	Operators narrowFilter, broadenFilter, and refocusFilter	
	for score filters \dots	32
3.14	Operator rejoin	33
3.15	Operators unrelate and untarget	33
3.16	Operator useAsCube	34

58 LIST OF TABLES

4.1	Navigation guard expressions concerning granularity levels (as	
	denotes a non-comparative and cas a comparative analysis	
	situation)	38
4.2	Navigation guard expressions concerning dice levels (as de-	
	notes a non-comparative and cas a comparative analysis situ-	
	ation)	39
4.3	Navigation guard expressions concerning dice nodes (as de-	
	notes a non-comparative and cas a comparative analysis situ-	
	ation)	40
4.4	Navigation guard expressions examining next and previous	
	dice node (as denotes a non-comparative and cas a compara-	
	tive analysis situation)	41
4.5	Navigation guard expressions examining slice conditions, filter	
	conditions, score filter conditions, and whether a result set is	
	empty (as denotes a non-comparative and cas a comparative	
	analysis situation)	42