Integrated Tool Support for the Datawarehouse Lifecycle

Elton Manoku
e.manoku@ft.han.nl
HAN University
Business and Competence Center
P.O. Box 2217
NL-6802 CE Arnhem
The Netherlands
Phone: +31(0) 26 365 81 52
Fax: +31(0) 26 365 81 26

Guido Bakema
guido.bakema@wxs.nl
HAN University
Informatics Communication Academy
P.O. Box 2217
NL-6802 CE Arnhem
The Netherlands
Phone: +31(0) 26 365 81 52
Fax: +31(0) 26 365 81 26

Abstract. In operational practice, datawarehouses are big corporate databases that are continuously under development. This implies a dynamic increase of the complexity of the data. For controlling this complexity, a conceptual model driven approach is recommended in order to guarantee that the relation to the business environment can be validated at any moment. To keep maintainable the connection between the conceptual model and the logical/physical aspects of the datawarehouse and related data marts, bridges are needed that can provide the needed model-to-model conversion at any desired moment. To achieve this, a bridge-toolset is designed and developed which is based on a single point of definition metadata philosophy. The philosophy and the way of working are developed in a research project at HAN University in The Netherlands in co-operation with the Dutch system house Atos Origin to overcome datawarehouse life cycle maintenance problems in the Royal Dutch Airlines (KLM).

1 Introduction

Section 1.1 gives a short introduction to the conceptual modeling method FCO-IM (Fully Communication Oriented Information Modeling) used for constructing the conceptual layer of a datawarehouse in the form of an FCO-IM information grammar. The conceptual layer is the starting point for all further data models playing a role in the datawarehouse lifecycle. Section 1.2 gives an impression of the transformation of these conceptual FCO-IM information models to the logical and physical layers.

1.1 Introduction to Fully Communication Oriented Information Modeling

In this short introduction, the main concepts and terminology of FCO-IM [1,2] are only touched upon. FCO-IM is a fact-based information modeling method that originates from NIAM [3]. The analysis takes place in a dialogue between the analyst and an expert user, who is requested to verbalize facts and express them through fact stating sentences (fact expressions). These fact expressions are classified and qualified and the type level result with constraints added is called an FCO-IM information grammar that can be visualized as an FCO-IM information grammar diagram. The classification and qualification is, apart from extending the soft semantics (predicates and type level naming), the main source for hard semantics (structural and integrity aspects of the model) for the information grammar. More hard semantics are added in the form of constraints (like uniqueness constraints, total role constraints and so on). The resulting information grammar (IG) is stored in a standardized FCO-IM repository and presented in one or more related information grammar diagrams (IGD's) that are self-synchronizing auto-visualizations of the repository population.

In the following Floors & Rooms example (see figure 1) elementary facts are considered. Using elementary fact expressions has many benefits. The most important one is that it guarantees redundancy freeness. The resulting information grammar is called an elementary information grammar (El-IG).

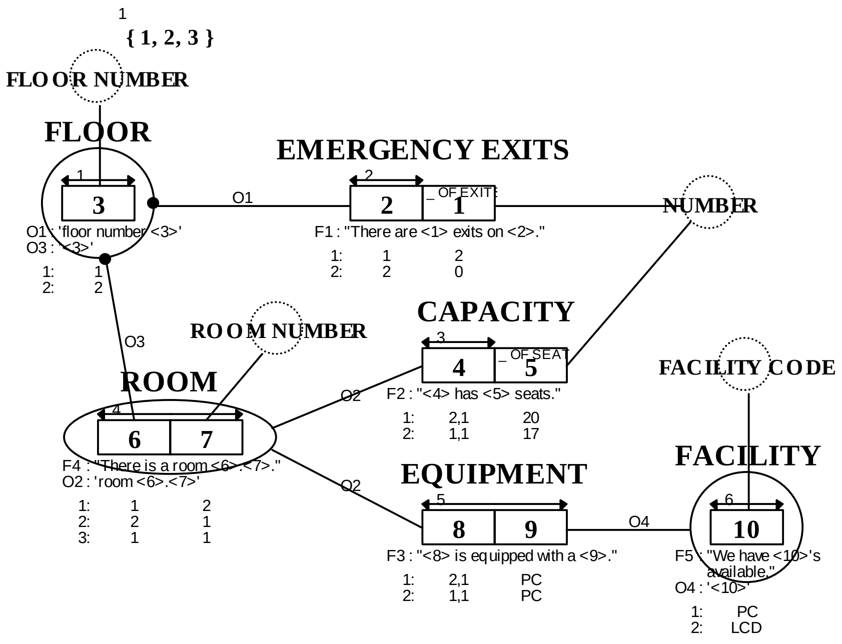
Figure 1: Floors & Rooms sample fact expressions

"Floor 1 exists."
"Floor 2 exists."
"There are 2 emergency exits on floor number 1."
"There are 0 emergency exits on floor number 2."
"There is a room 2.1."
"There is a room 1.1."
"There is a room 1.2."

"Room 2.1 has 20 seats."
"Room 1.1 has 17 seats."
"Room 2.1 is equipped with a PC."
"Room 1.1 is equipped with a PC."
"We have PC's available."
"We have LCD's available."

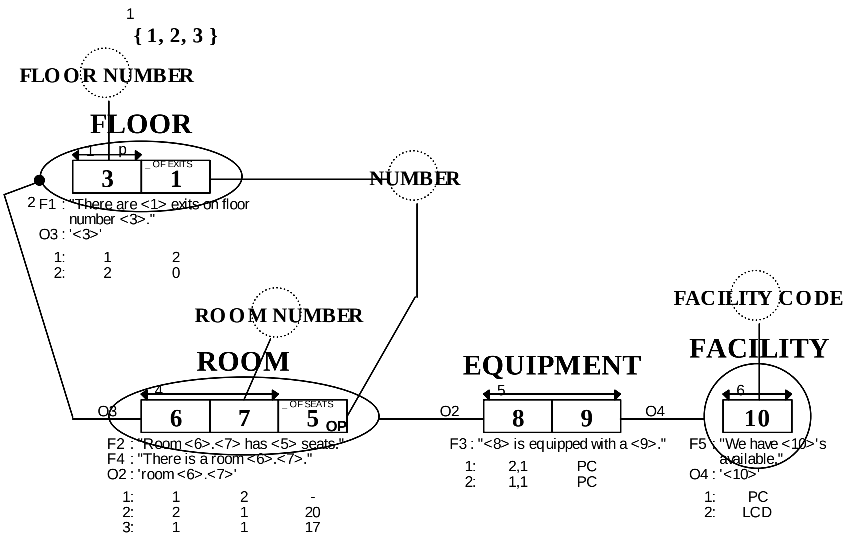
The diagram (El-IGD) of the resulting elementary information grammar (El-IG) is shown in figure 2.

Figure 2: Floors & Rooms El-IGD



Applying the FCO-IM Group & Reduce algorithm to the elementary information grammar (El-IG) results in a grouped and reduced information grammar (GR-IG). See figure 2 for the corresponding diagram (GR-IGD).

Figure 3: Floors & Rooms GR-IGD

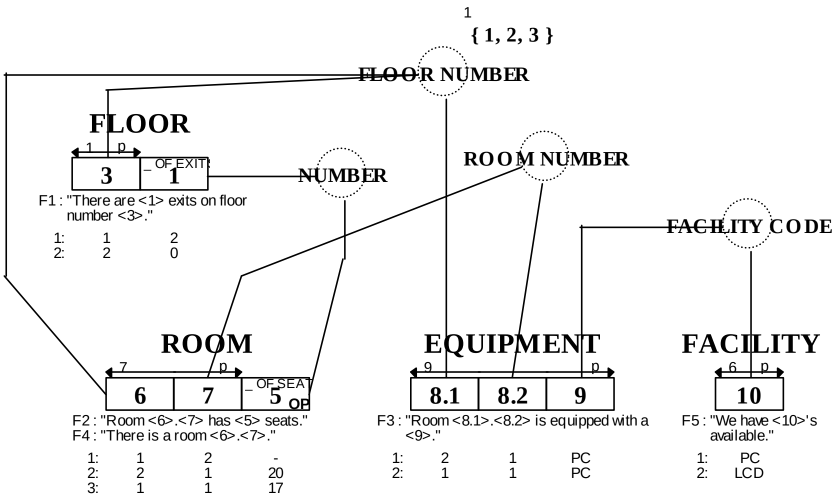

{ 1, 2, 3 }
FLOOR NUMBER
FLOOR
NUMBER
_OF EXITS
F1 : "There are <1> exits on floor number <3>."
O3 : '<3>'
ROOM NUMBER
FACILITY CODE
ROOM
EQUIPMENT
FACILITY
_OF SEATS
OP
O3
O2
O4
F2 : "Room <6>.<7> has <5> seats."
F4 : "There is a room <6>.<7>."
O2 : 'room <6>.<7>'
F3 : "<8> is equipped with a <9>."
F5 : "We have <10>'s available."
O4 : '<10>'

The Group & Reduce algorithm brings fact types together as much as possible without introducing redundancy. In other words, starting from the elementary information grammar, this process gives a normalized model of the information grammar with a minimum number of fact types. From this grouped and reduced information grammar, fact sentences can still be reproduced.

Applying also (in between or afterwards) the Lexicalize algorithm results in a grouped, lexicalized and reduced information grammar (GLR-IG). See figure 4 for the diagram (GLR-IGD).

From all these FCO-IM information grammars (EL-IG, GR-IG, GLR-IG) the fact sentences can be regenerated.

Figure 4: Floors & Rooms GLR-IGD


{ 1, 2, 3 }
FLOOR NUMBER
FLOOR
NUMBER
ROOM NUMBER
FACILITY CODE
F1 : "There are <1> exits on floor number <3>."
ROOM
EQUIPMENT
FACILITY
F2 : "Room <6>.<7> has <5> seats."
F4 : "There is a room <6>.<7>."
F3 : "Room <8.1>.<8.2> is equipped with a <9>."
F5 : "We have <10>'s available."

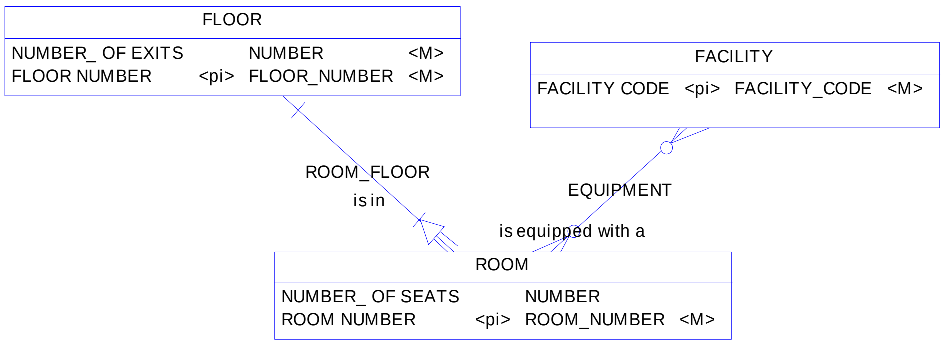
To support this way of working with different co-existing and related FCO-IM models and keeping synchronization between them, a consistent repository-based approach is used. Every FCO-IM model (El-IG, GR-IG, GLR-IG) exists in a generic repository and the GLR algorithms are just reading and updating this FCO-IM repository that is used by FCO-IM supporting CASE-tools like **FCO-IM Casetool**[4] and **CaseTalk**[5], which are widely in use in the Netherlands in the academic world and in industry.

1.2 Unmasking the Conceptual Models

Essentially, the GLR-IGD of figure 4 already is a Relational Schema in BCNF. Database administrators might like to see that in a more familiar representation. The GLR-IGD must, so to speak, drop its FCO-IM mask and present itself more openly as a Relational Schema. Apart from the GLR algorithms, also the Relational Model (RM) unmasking algorithm is implemented in FCO-IM supporting tools. Because all soft semantics are stored, fact type expressions (i.e. fact expressions at type level) can be generated along with the resulting Relational Schema as well, providing hard semantics as well soft semantics.

The GR-IGD of figure 3 is essentially an Entity-Relationship diagram (ERD). The ER demasqué algorithm converts a GR-IGD into an ERD in a more familiar Entity-Relationship notation. See figure 5 where the Information Engineering syntax is used.

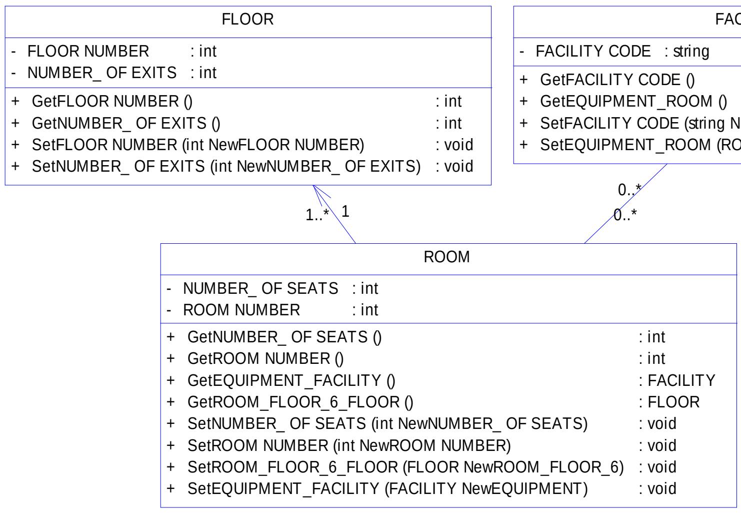
Figure 5: Floors & Rooms ERD


FLOOR
NUMBER_ OF EXITS NUMBER <M>
FLOOR NUMBER <pi> FLOOR_NUMBER <M>
FACILITY
FACILITY CODE <pi> FACILITY_CODE <M>
ROOM_FLOOR
is in
EQUIPMENT
is equipped with a
ROOM
NUMBER_ OF SEATS NUMBER
ROOM NUMBER <pi> ROOM_NUMBER <M>

The UML unmasking algorithm converts a GR-IGD into an UML Class Diagram. See figure 6.

In both cases the sentences cannot be presented any more along with the diagrams, but they are still present in the form of comments or descriptions in structure elements, nor can a population be presented. In the case of a Class Diagram, an object diagram can present the population. Within ER and UML tools, then, a physical model and Relational and OO data definition scripts can easily be generated for Relational and OO platforms.

Figure 6: Floors & Rooms UML Class Diagram


FLOOR
- FLOOR NUMBER : int
- NUMBER_ OF EXITS : int
+ GetFLOOR NUMBER () : int
+ GetNUMBER_ OF EXITS () : int
+ SetFLOOR NUMBER (int NewFLOOR NUMBER) : void
+ SetNUMBER_ OF EXITS (int NewNUMBER_ OF EXITS) : void
FACILITY
- FACILITY CODE : string
+ GetFACILITY CODE () : string
+ GetEQUIPMENT_ROOM () : ROOM
+ SetFACILITY CODE (string NewFACILITY CODE) : void
+ SetEQUIPMENT_ROOM (ROOM NewEQUIPMENT) : void
1..*
1
0..*
0..*
ROOM
- NUMBER_ OF SEATS : int
- ROOM NUMBER : int
+ GetNUMBER_ OF SEATS () : int
+ GetROOM NUMBER () : int
+ GetEQUIPMENT_FACILITY () : FACILITY
+ GetROOM_FLOOR_6_FLOOR () : FLOOR
+ SetNUMBER_ OF SEATS (int NewNUMBER_ OF SEATS) : void
+ SetROOM NUMBER (int NewROOM NUMBER) : void
+ SetROOM_FLOOR_6_FLOOR (FLOOR NewROOM_FLOOR_6) : void
+ SetEQUIPMENT_FACILITY (FACILITY NewEQUIPMENT) : void

2 An Integrated Toolset for Datawarehouse Design

In section 2.1 the datawarehouse life cycle is characterized and challenges are discussed. Section 2.2 introduces a Metadata Framework that was designed to master these challenges. Section 2.3 presents a supporting tool-set that can help to deal with the challenges in big datawarehouse projects. The most important one is a tool for automated conversion of normalized models into dimensional models. Section 2.4 ends with the general architecture of the supporting tool-set.

2.1 Challenges from the Datawarehouse World

The datawarehouse lifecycle process starts with interviewing future datawarehouse users and gathering information about source systems. This is followed by the analysis and the design of the datawarehouse. The next stage is the implementation, including designing and implementing the ETL process, data marts design and generation and development of OLAP applications. Eventually these are put in use in order to provide the desired business intelligence. This process looks rather straightforward, but in general it is not. There are many challenges. Finding equilibrium between user demands and the source systems involves quantitative (many fact types) and qualitative (integrating domain area’s) complexity. The transformation towards dimensional models is, from the design point of view, another complex process. These challenges are difficult to master. In general the development of big corporate datawarehouses is an evolutionary process: the datawarehouse is continuously developing, with all version-to-version management challenges.

Let’s suppose that the data architects succeed to come up with a first version of a data model for the datawarehouse that can be related to the source systems and sufficiently meets the user requirements. The danger exists that after the data model is retrieved and further versions are developed, the connection between the initial user requirements and the definitive data model gets lost. This danger rises also when a physical model is retrieved and again if dimensional models are derived. At the end it might be a surprise that the result is no longer something that meets the user requirements. This danger can be avoided if all model-to-model conversions are made algorithmically, keeping alive, during these conversions, the correspondence between user requirements and data models.

2.2 A Framework for Metadata Management

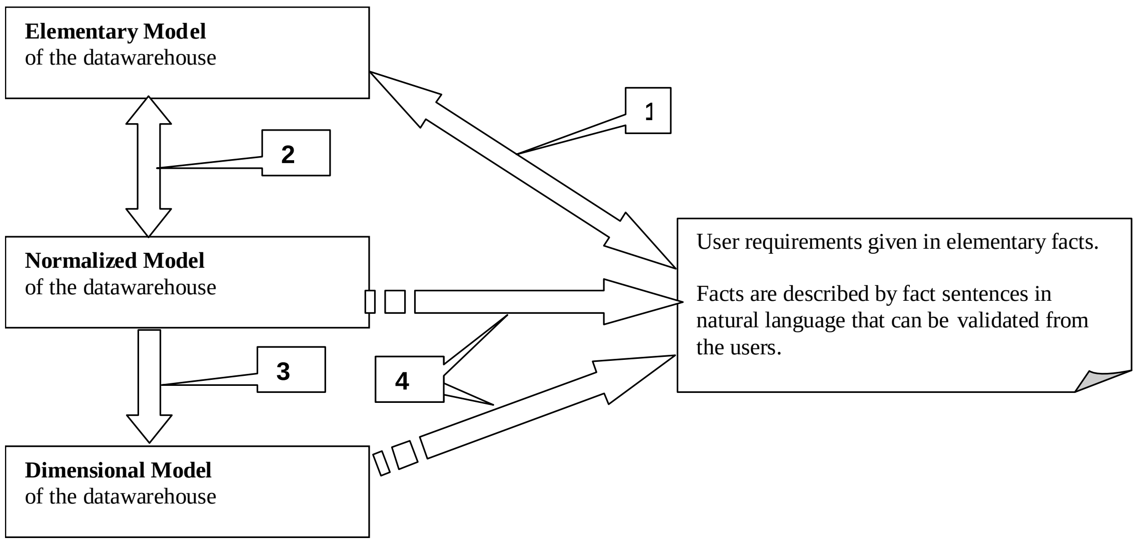
In this section a framework is presented for meeting the above-mentioned challenges during datawarehouse projects by means of proper metadata management. This Metadata Framework[6,7] tries to conquer the version-to-version complexity and the dangers related to the model-to-model transformation by giving the metadata of the elementary FCO-IM model of the datawarehouse a single-point-of-definition status. All other models are derived from that in an automated way. The idea is a layering of the datawarehouse lifecycle based on different phases of the datawarehouse development. Jumping from one development layer to another consists in transforming the metadata at the conceptual level, whilst continuously being able to convert it on demand to desired logical and physical platforms: Entity-Relationship, UML and/or Relational.

These datawarehouse development layers include **Elementary Models**, **Normalized Models** and equivalent **Dimensional Models** in the form of families of stars with conformal dimensions[8] that can easily be split up into a collection of separate data marts, consisting of just one or a few related stars. All of this can be presented at a conceptual (i.e. FCO-IM style models), logical (i.e. Entity-Relationship models and/or UML class diagrams) and physical (i.e. Relational Schema's) level and as such be imported in logical and physical level tools.

This way of working also guarantees maintaining a close relationship between hard and soft semantics at any moment, which is considered essential for running datawarehouse projects successfully. Datawarehouse users can validate the present version of the conceptual models at any moment and assure the datawarehouse team that they are working in the right way. On the other hand, logical and physical models fulfilling the validated business requirements and the technical aspects of datawarehouses and data marts can be generated instantaneously, even, if desired, with preservation of all soft semantics together with the hard semantics.

A simplified picture of the Metadata Framework, focusing on the layering from the data model perspective, is given in figure 7.

Figure 7: Simplified picture of the Metadata Framework


Elementary Model
of the datawarehouse
Normalized Model
of the datawarehouse
Dimensional Model
of the datawarehouse
1
2
3
4
User requirements given in elementary facts.
Facts are described by fact sentences in natural language that can be validated from the users.

The starting point for modeling the datawarehouse is a set of elementary facts covering all user needs concerning the datawarehouse. These elementary facts are the source for the Elementary Model. Arrow 1 shows that the **Elementary Model** can be retrieved from analyzing elementary fact expressions and the elementary fact expressions can be regenerated from the Elementary Model.

The **Normalized Model** of the datawarehouse is a more compact view of the elementary facts. The elementary fact types are grouped together as much as possible without introducing redundancy, the resulting FCO-IM information grammar is equivalent with an Entity-Relationship Model or Relational Schema in at least BCNF. See section 1.2. Arrow 2 shows that the Normalized Model can be retrieved from the Elementary Model by applying the GR algorithm or the GLR algorithm (see section 1.1). If the facultative R (reducing) part of the algorithm it is not used, the Conceptual Model can again be retrieved from the Normalized Model.

The **Dimensional Model** of the datawarehouse consists of elementary fact types further grouped (actually overgrouped) for getting a denormalized dimensional view in the form of a dimensional FCO-IM information grammar (family of stars in FCO-IM style) that can be unmasked to get an equivalent Entity-Relationship Model or Relational Schema as well. Arrow 3 shows that the Dimensional Model can be retrieved from the Normalized Model by applying the StarBridge algorithm[9,10,11], based on the covering forest theorem[12]. In the Dimensional Model redundancy is introduced in such a way that candidate Fact Tables and related Dimension Tables are indicated. Arrow 4 shows that domain experts at any desired moment can retrieve the elementary fact expressions in natural language for validation purposes.

2.3 Supporting Tool-set

The Metadata Framework is supported by an FCO-IM based modeling and model-to-model conversion tool-set. From the point of view of functionality, the tools are of two different types:

a. Decision driven modeling and model-to-model conversion at the conceptual level. This kind of tools is used to transform a conceptual model into another conceptual model, more or less based on decisions of the analyst. The relevant tools of this type used for support of the Metadata Framework way of working are:
   - **CaseTalk**. This is an FCO-IM modeling tool used by the analysts to translate the facts into diagrams by adding structure and integrity to fact expressions and bringing them on type level in an Elementary Model (El-IG). The tool is completely repository based and offers in-repository Group, Lexicalize and Reduce algorithms with all resulting conceptual diagrams and the demasqué algorithm for the presentation of a Normalized Model (GLR-IG) in more familiar Relational style. See section 1.
   - **StarBridge**. This FCO-IM based tool supports the analyst for retrieving a Dimensional Model (D-IG) and as such, like **CaseTalk**, is an important conversion tool to support the Metadata Framework layers at the conceptual level. This tool takes as input a Normalized Model (FCO-IM GR-IGD) and outputs a denormalized Dimensional Model for the datawarehouse. Recently the FCO-IM repository was extended in order to be able to store at the conceptual level dimensional datawarehouse models (FCO-IM D-IG's) as well and a conversion algorithm was specified that helps the analyst to convert a conceptual normalized model (FCO-IM GR-IG) to an equivalent dimensional Model (FCO-IM D-IG). The tool implements the StarBridge algorithm that consists of several distinct steps. See figure 8.

b. Supporting straightforward model-to-model demasqué conversions. These tools are used to transport the conceptual metadata to the logical/physical platforms. For support of the Metadata Framework, the following tools are available:
   - **ER Bridge**. This tool provides a bridge between the FCO-IM world and the Entity-Relationship world. It exports conceptual models (Normalized Models and Dimensional Models) to the logical/physical world of ER tools. It realizes this by using repository-to-repository transformation: export and conversion of the metadata of an FCO-IM Model (GR-IG) to an intermediate Entity-Relationship Repository and export from that into ER tools.
   - **UML Bridge**. This tool provides a bridge between the FCO-IM world and the UML world (Class Diagrams). It realizes by using repository-to-repository transformation, the export of an FCO-IM Model (GR-IG) to an intermediate Class Diagram Repository and export from that into UML tools.

Figure 8: StarBridge tool architecture

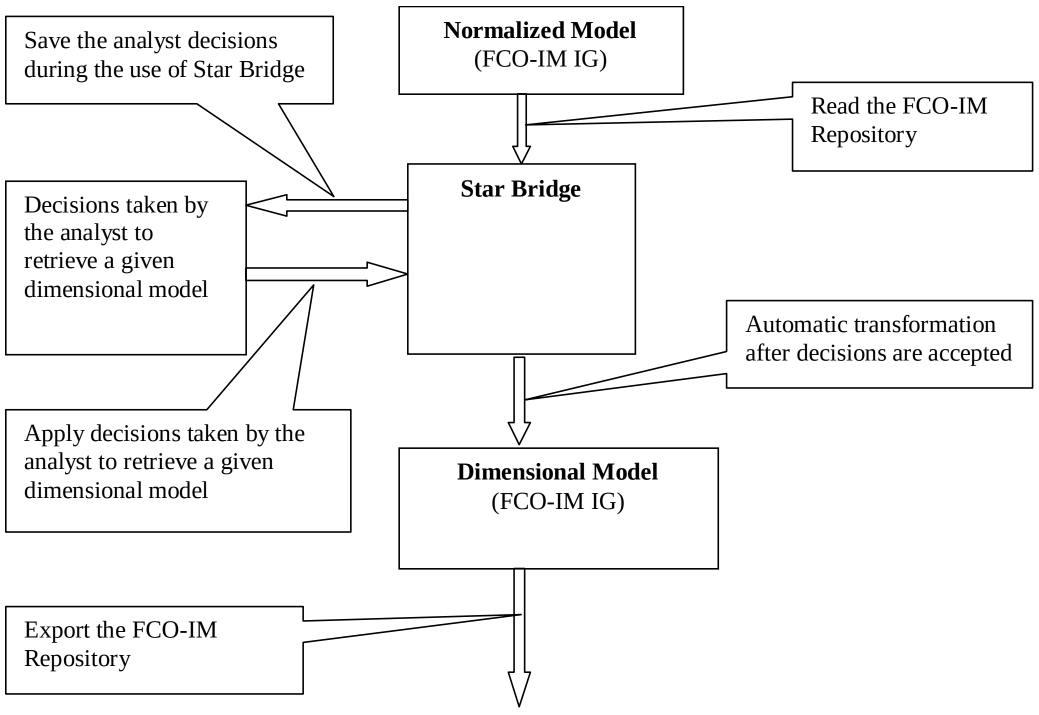
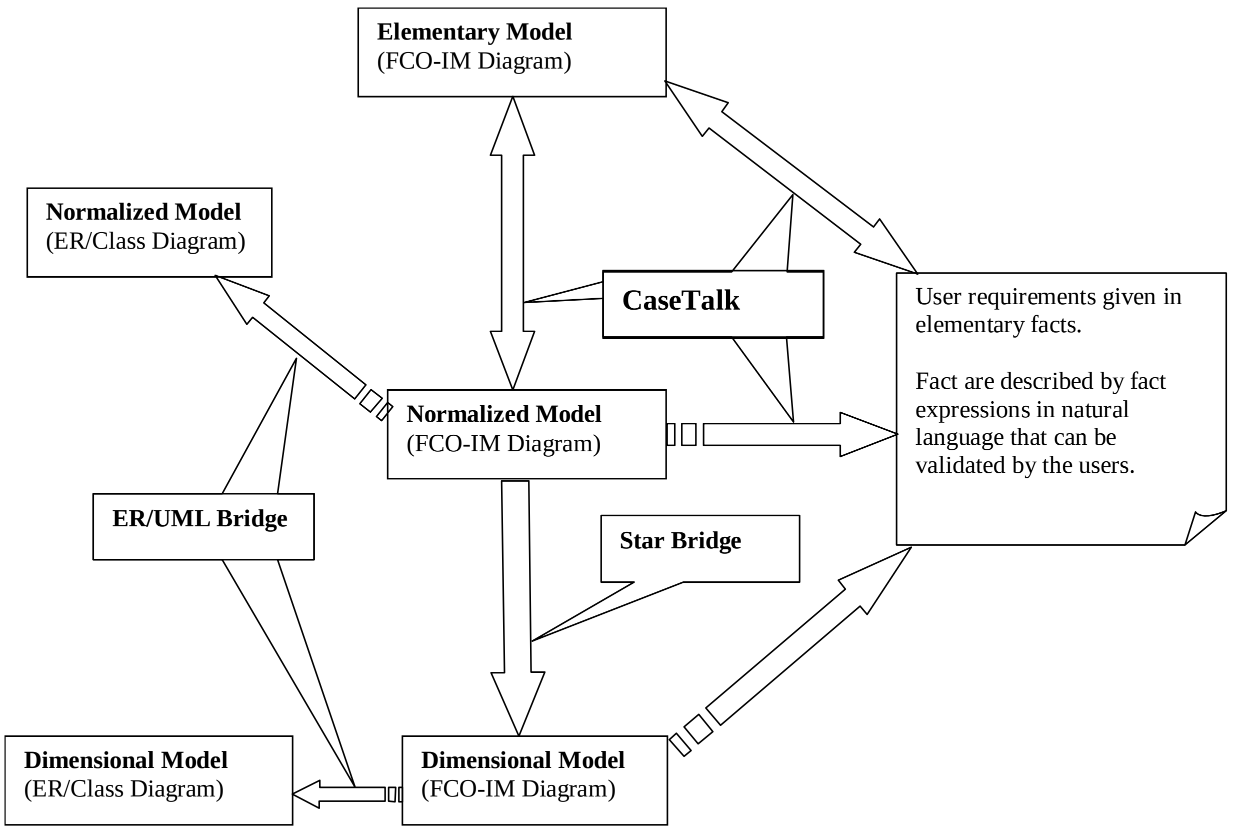

Save the analyst decisions during the use of Star Bridge
Normalized Model (FCO-IM IG)
Read the FCO-IM Repository
Star Bridge
Decisions taken by the analyst to retrieve a given dimensional model
Automatic transformation after decisions are accepted
Apply decisions taken by the analyst to retrieve a given dimensional model
Dimensional Model (FCO-IM IG)
Export the FCO-IM Repository

Figure 9 shows how the tools are used in an integrated way in the Metadata Framework.

Figure 9: Integrated Tool Support for the Metadata Framework



2.4 Tool Architecture

Even though the conversion tools are implementing different algorithms, all conversion tools are based on in-repository or repository-to-repository transformations. At the moment all repositories are in Relational format and the in-repositories or repository-to-repository transformations are made by Structured Query Language (SQL). **ER Bridge**, **UML Bridge** and **StarBridge** are modules integrated in one **FCO-IM Bridge** tool. They receive an appropriate copy of the populated Relational FCO-IM repository from the FCO-IM modeling tool **CaseTalk**.

As a consequence, the tools can be conceived as sets of queries running in a given order, and developing this kind of tools means thinking about SQL-instructions that carry out tasks. Each instruction has properties related to them such as name of instruction, type, body, what step of the algorithm it is involved in, order of execution, etc. These properties are stored in a repository along with the instructions themselves and their descriptions. This architecture makes tool implementation easier and helps a lot for tool maintenance. Here, the idea of storage of the hard and soft semantics of the algorithm in a consistent and redundant free way is applied again.

There are two groups of reasons for choosing a repository-based architecture for the transformations as well:

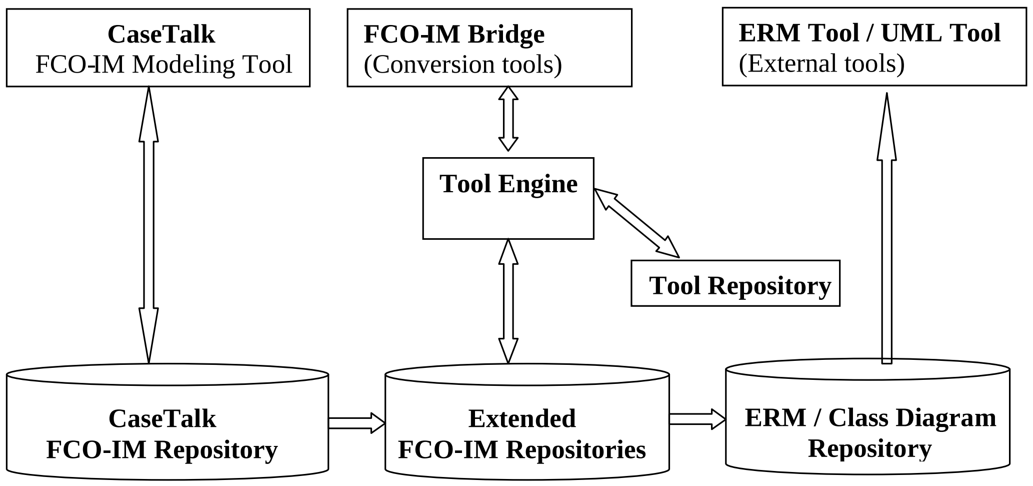
a) Reasons related to the algorithms applied in the tools. What makes conversion tools different from each other is the algorithm implemented by them. Regarding to the implemented algorithm some reasons to use a repository is:
   - Single source of algorithm logic.
   - Single source of algorithm description.
   - Automatic generation of the implemented algorithm documentation.
   - Easy maintenance of the implemented algorithm.
   - Generation of code for other programming language platforms.
   - The algorithm is not part of the code and can be easily changed and maintained.

b) Reasons related to the User Interface of the tool:
   - To get an efficient tool a good terminology is required. By terminology is meant: what to call the different operations within the tool, the different steps, the notations for parts of the algorithms (‘Dimension’, ‘Do not reduce’, etc.), the messages, the captions, etc. Most probably this differs from one language to another. For the best notation, the best strategy is again to interview many people. If this kind of information is stored in a repository, every analyst can select his own notation.
   - For parts of the interface that are highly dynamic, like the order of interface steps, it is convenient to keep this information in repository. From the repository, it is much easier to impose rules on order of steps.

For running the tool logic against the tool repository, an engine is needed. This engine depends on the tool's repository structure, but has nothing to do with the algorithm stored in the tool repository. The engine is a set of functions and procedures that makes possible the communication between the tool logic and the tool repository and handles the information found in the tool repository. The engine must handle the information on the algorithm stored in the tool repository, and the tool interface. In designing of such an engine the main concern is:
- Handle any possible case that can be found in the algorithms.
- Retrieve / update terminology used in the tool.
- Have redundant free code as much as possible.
- Use dynamic structures of functions making them reusable for different cases.

See figure 10 for the overall architecture of the tool-set.

Figure 10: The overall Architecture of the Tool-set



The **CaseTalk** modeling tool uses the basic FCO-IM Repository, in which the following conceptual models are stored: elementary and grouped & reduced FCO-IM information grammars. The ER Tool Repository and UML-Class Diagram Tool Repository are external ER or UML Tool repositories (CA Allfusion ERwin 4.1, Sybase PowerDesigner 9) where the exported models can be stored.

At the moment, the exchange of models between the repositories of the tools is based on different techniques.
- The export of the models from **CaseTalk** to **FCO-IM Bridge** is Relational format based.
- The export of the models from **FCO-IM Bridge** to the external ER and UML tools is based on XML document exchange (ERwin 4.1) and VBScript generation (PowerDesigner 9). For the techniques that can be used, it is of course relevant what the external tool offers.

In order to achieve independence from the external ER tools, a Metadata Repository Model for storing ER Models is used that also supports the storage of alternative keys. This model is physically implemented in the form of views on top of the extended FCO-IM Repository. In the near future an XML Schema format will be used for supporting this exchange. The same goes for UML-Class Diagrams.

3 The Approach in Practice

This section shows how the outlined approach is used in practice. Section 3.1 gives a few characteristics of the KLM Passage datawarehouse project. In this long term project the described approach is used and developed further, and the project serves as an operational test environment for the tool-set that is continuously developing. In section 3.2 it is demonstrated, using only a small part of the KLM Passage Model, how the StarBridge algorithm works in practice.

3.1 KLM Passage Model

The KLM Passage Model is a corporate datawarehouse data model for the passenger division in Royal Dutch Airlines (KLM). Data are extracted from 5 different source systems and from the Internet.

The modeling tools used in the KLM Passage project are: **CaseTalk**, **FCO-IM Bridge** and **ERwin**.

The number of fact types in the conceptual Elementary Model (an FCO-IM El-IG) is more than 600. The logical Normalized Model (an ERD) that is generated from the conceptual Normalized Model has more than 180 entity types. The physical Dimensional Model (family of stars with conformal dimensions) generated from the conceptual Dimensional Model has 16 fact tables and about 37 dimension tables, whilst 177 relationships refer from the fact tables to the dimension tables. At the moment about 13 separate data marts (each consisting of 1 or 2 stars) are loaded on a weekly or monthly basis.

3.2 The Forecast Sub-model

From the KLM Passage Model the Forecast Sub-model is used here to demonstratehow the StarBridge algorithm works. Only the modeling part of the datawarehouse lifecycle is considered in this example. For simplicity (not overloading figures) and privacy reasons, the population will not be part of diagrams.

Suppose that the analysts, after interviewing potential datawarehouse users and taking into account what data can be provided by the source systems, come up with a set of fact expressions shown in figure 11. To be sure that no redundancy is present, the fact expressions are elementary. They are also formulated in the desired grain.

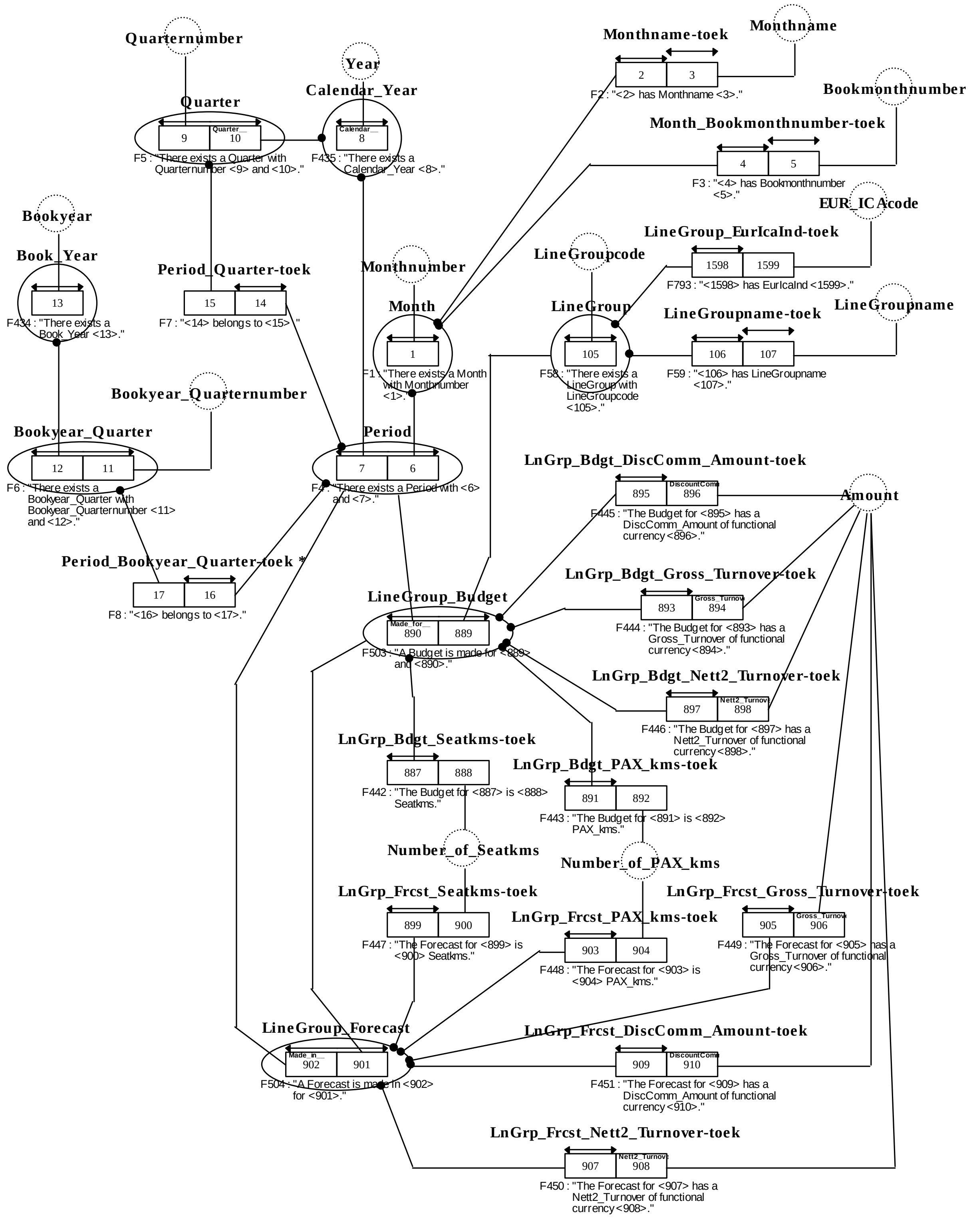
Figure 11: Sample set of fact expressions for the Forecast Sub-model of the KLM Passage Model

"There exists a Month with Monthnumber 11."
"There exists a Month with Monthnumber 04."
"Month 04 has Monthname April."
"Month 01 has Bookmonthnumber 10."
"Month 04 has Bookmonthnumber 01."
"Period (01, 1998) belongs to Bookyear_Quarter (4, 1997/1998)."
….
"There exists a LineGroup with LineGroupcode A."
"LineGroup A has LineGroupname Europe."
"A Budget is made for LineGroup A and Period (04, 1998)."
"A Budget is made for LineGroup A and Period (03, 1998)."
….
"The Budget for LineGroup A and Period (04, 1998) has a DiscComm_Amount of functional currency 5."
"The Budget for LineGroup A and Period (03, 1998) has a DiscComm_Amount of functional currency 6."
"The Budget for LineGroup A and Period (04, 1998) has a Gross_Turnover of functional currency 25."
"The Budget for LineGroup A and Period (03, 1998) has a Gross_Turnover of functional currency 23."
….

The next step is the classification and qualification of these fact expressions. This step is executed by using the FCO-IM modeling tool **CaseTalk**. Constraints are added by further interviewing domain experts. Hard and soft semantics are automatically stored in the **CaseTalk** FCO-IM Repository. From the resulting conceptual Elementary Model (see figure 12) fact sentences can be regenerated for user validation.

To obtain a Normalized Model in conceptual form, these fact types are grouped together without losing the granularity that is expressed in the fact expressions. This step is also performed by **CaseTalk** by applying the Group & Reduce algorithm to the Elementary Model. See figure 12.

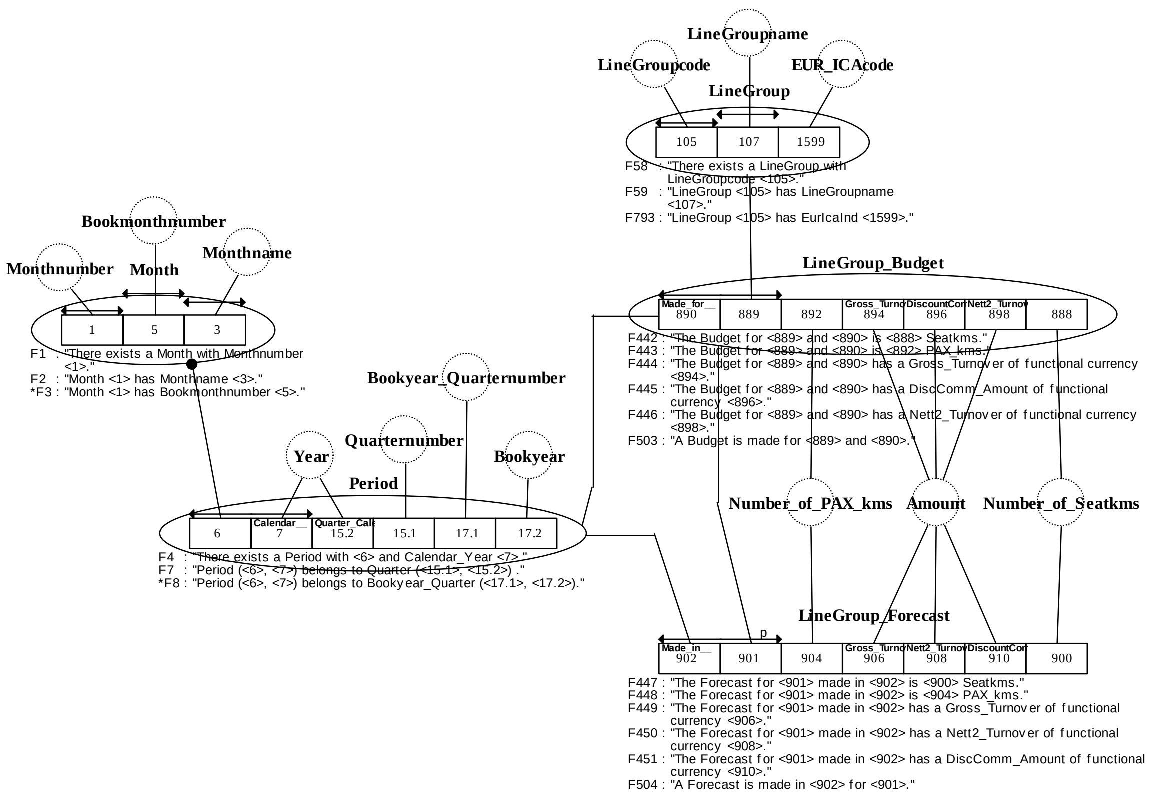
Figure 12: The conceptual Elementary Forecast Sub-Model of the KLM Passage Model


Quarternumber
Year
Monthname-toek
Monthname
F2 : "<2> has Monthname <3>."
Bookmonthnumber
Quarter
Calendar_Year
F5 : "There exists a Quarter with Quarternumber <9> and <10>."
F435 : "There exists a Calendar_Year <8>."
Month_Bookmonthnumber-toek
F3 : "<4> has Bookmonthnumber <5>."
EUR_ICAcode
Bookyear
Book_Year
F434 : "There exists a Book_Year <13>."
Period_Quarter-toek
F7 : "<14> belongs to <15>."
Monthnumber
Month
F1 : "There exists a Month with Monthnumber <1>."
LineGroupcode
LineGroup
F58 : "There exists a LineGroup with LineGroupcode <105>."
LineGroup_EurIcaInd-toek
F793 : "<1598> has EurIcaInd <1599>."
LineGroupname-toek
LineGroupname
F59 : "<106> has LineGroupname <107>."
Bookyear_Quarternumber
Bookyear_Quarter
F6 : "There exists a Bookyear_Quarter with Bookyear_Quarternumber <11> and <12>."
Period
F4 : "There exists a Period with <6> and <7>."
LnGrp_Bdgt_DiscComm_Amount-toek
F445 : "The Budget for <895> has a DiscComm_Amount of functional currency <896>."
Amount
Period_Bookyear_Quarter-toek *
F8 : "<16> belongs to <17>."
LnGrp_Bdgt_Gross_Turnover-toek
F444 : "The Budget for <893> has a Gross_Turnover of functional currency <894>."
LineGroup_Budget
F503 : "A Budget is made for <889> and <890>."
LnGrp_Bdgt_Nett2_Turnover-toek
F446 : "The Budget for <897> has a Nett2_Turnover of functional currency <898>."
LnGrp_Bdgt_Seatkms-toek
F442 : "The Budget for <887> is <888> Seatkms."
LnGrp_Bdgt_PAX_kms-toek
F443 : "The Budget for <891> is <892> PAX_kms."
Number_of_Seatkms
Number_of_PAX_kms
LnGrp_Frcst_Seatkms-toek
F447 : "The Forecast for <899> is <900> Seatkms."
LnGrp_Frcst_PAX_kms-toek
F448 : "The Forecast for <903> is <904> PAX_kms."
LnGrp_Frcst_Gross_Turnover-toek
F449 : "The Forecast for <905> has a Gross_Turnover of functional currency <906>."
LineGroup_Forecast
F504 : "A Forecast is made in <902> for <901>."
LnGrp_Frcst_DiscComm_Amount-toek
F451 : "The Forecast for <909> has a DiscComm_Amount of functional currency <910>."
LnGrp_Frcst_Nett2_Turnover-toek
F450 : "The Forecast for <907> has a Nett2_Turnover of functional currency <908>."

In the Normalized Model diagram the entity types LineGroup_Forecast, Period, LineGroup_Budget, Month and LineGroup are present.

In figure 13 the Normalized datawarehouse Model is still in its conceptual shape. From the Normalized Model the fact expressions can still be retrieved for validation purposes. To go into the logical / physical level (towards implementation) the **ER Bridge** module of the **FCO-IM Bridge** tool can generate an ERD that can be imported in **ERwin**.

Figure 13: The Normalized Forecast Sub-Model of the KLM Passage Model in conceptual form



F58 : "There exists a LineGroup with LineGroupcode <105>."
F59 : "LineGroup <105> has LineGroupname <107>."
F793 : "LineGroup <105> has EurIcaInd <1599>."

F1 : "There exists a Month with Monthnumber <1>."
F2 : "Month <1> has Monthname <3>."
*F3 : "Month <1> has Bookmonthnumber <5>."

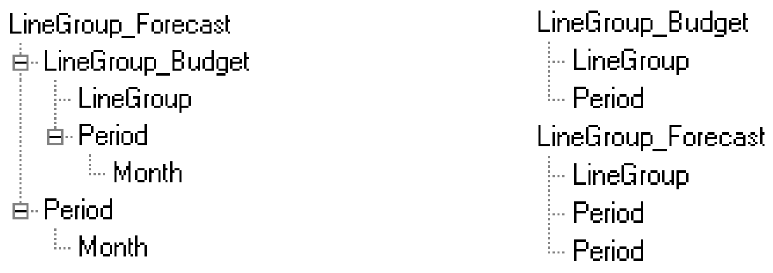
F442 : "The Budget for <889> and <890> is <888> Seatkms."
F443 : "The Budget for <889> and <890> is <892> PAX_kms."
F444 : "The Budget for <889> and <890> has a Gross_Turnover of functional currency <894>."
F445 : "The Budget for <889> and <890> has a DiscComm_Amount of functional currency <896>."
F446 : "The Budget for <889> and <890> has a Nett2_Turnover of functional currency <898>."
F503 : "A Budget is made for <889> and <890>."

F4 : "There exists a Period with <6> and Calendar_Year <7>."
F7 : "Period (<6>, <7>) belongs to Quarter (<15.1>, <15.2>) ."
*F8 : "Period (<6>, <7>) belongs to Bookyear_Quarter (<17.1>, <17.2>)."

F447 : "The Forecast for <901> made in <902> is <900> Seatkms."
F448 : "The Forecast for <901> made in <902> is <904> PAX_kms."
F449 : "The Forecast for <901> made in <902> has a Gross_Turnover of functional currency <906>."
F450 : "The Forecast for <901> made in <902> has a Nett2_Turnover of functional currency <908>."
F451 : "The Forecast for <901> made in <902> has a DiscComm_Amount of functional currency <910>."
F504 : "A Forecast is made in <902> for <901>."

The next step is the conversion of the Normalized Model towards a Dimensional Model. For this purpose, the **StarBridge** module of the **FCO-IM Bridge** tool is used. It helps the analyst by proposing options (candidate fact tables and dimensions, providing conformity by splitting off a mini dimension or by introducing an aggregate dimension and so on) and it will automatically apply any decision that the analyst makes. It even helps by proposing proper decisions. In this example, the StarBridge algorithm would suggest that LineGroup_Forecast is a fact table and Period and LineGroup_Budget are dimensions. Month and LineGroup are snowflake dimensions, which the analyst can decide to denormalize later into respectively Period and LineGroup_Budget. In this case, the analyst by looking into the soft-semantics (fact expressions) associated to the facts that are grouped into entity types, realizes that LineGroup_Budget must be a fact table, too. After this decision, the LineGroup is a dimension. The tree diagram would then be as shown in left side of figure 14.

Figure 14: Trees of the Normalized Forecast Sub-Model before and after denormalizing



```
LineGroup_Forecast
⊟·· LineGroup_Budget
      ··· LineGroup
    ⊟·· Period
          ··· Month
⊟·· Period
      ··· Month
```

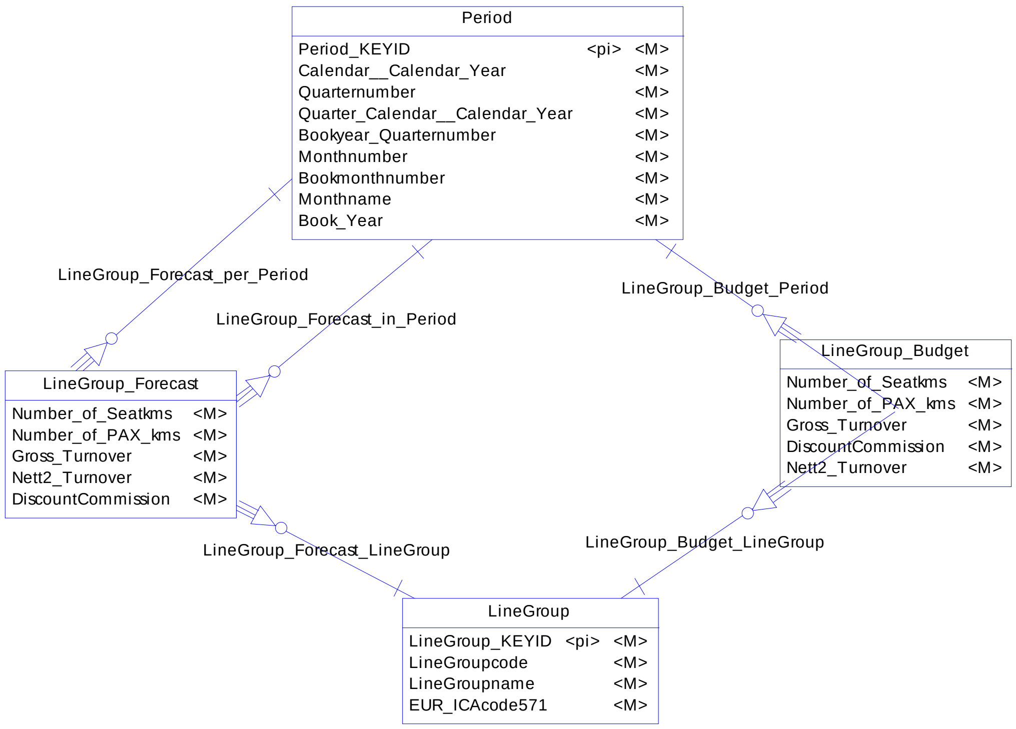
```
LineGroup_Budget
  ··· LineGroup
  ··· Period
LineGroup_Forecast
  ··· LineGroup
  ··· Period
  ··· Period
```

Only Month will be denormalized into Period. The definitive model has 2 fact tables with 2 conformal dimensions. See figure 15 right side. This model is stored in the extended FCO-IM Repository as a conceptual Dimensional Model. From this model the initial fact expressions can still be generated automatically for validation purposes.

The next step is towards a logical / physical Dimensional Model. For this purpose the **ER Bridge** module of the **FCO-IM Bridge** tool helps again, and the resulting logical Dimensional Model is imported into the ER tool **ERwin**. By exporting the Dimensional Model to an ER tool, the analyst benefits from the features these tools offer for the further implementation phase of the datawarehouse.

In **FCO-IM Bridge** the soft semantics are first transported from the extended FCO-IM Repository into the intermediate ER Repository and then exported to the chosen ER tool as entity types or attribute names and comments, relationship names and role descriptions. So, the soft semantics are not lost, but are still part of the model, but no longer in a structured shape. The Dimensinal Model in ER notation now looks as shown in figure 15. It shows two related stars with conformal dimensions.

Figure 15: Entity-Relationship diagram showing two related stars with conformal dimensions



References

1 G.P. Bakema, J.P.C. Zwart, H. van der Lek, *Fully Communication Oriented NIAM*, NIAM-ISDM 1994 Conference, Working Papers, pages L1-L35, Albuquerque, New Mexico (1994), www.FCO-IM.com

2 Guido Bakema, Jan Pieter Zwart, Harm van der Lek, *Volledig Communicatiegeoriënteerde Informatiemodellering*, tenHagenStam, 1996

3 G.M. Nijssen, T.A. Halpin, *Conceptual Schema and Relational Database Design: a fact oriented approach*, Prentice-Hall, 1989

4 *FCO-IM Casetool*, Ascaris Software & FCO-IM Consultancy, 1996/1997, www.FCO-IM.com

5 *CaseTalk*, Bommeljé Crompvoets and partners, 2002, www.CaseTalk.com

6 Peter Alons, *Single point of definition voor metadata*, Database Magazine, Dec 2000,www.FCO-IM.com

7 Peter Alons, *Beter modelleren begint op conceptueel niveau*, Database Magazine, Feb 2000,www.FCO-IM.com

8 Ralph Kimball, *The Data warehouse Toolkit*, John Wiley & sons, 1996

9 Harm van der Lek, *Op jacht naar de sterrren*, Database Magazine, April 2000

10 Jorg Janssens, Egi Rodriguez, *Extensions of FCO-IM*, HAN University masters thesis, Aug 1999

11 Rob Arntz, *Algorithmische transformatie van Conceptuele Modellen naar Stermodellen*, HAN University / Nijmegen university masters thesis, Aug 2000

12 Harm van der Lek, *Overdekkende Bos Stelling*, Database Magazine, Feb 2000