The i•com Tool for Intelligent Conceptual Modelling

Enrico Franconi and **Gary Ng**
Department of Computer Science,
University of Manchester, UK
{franconi|ngg}@cs.man.ac.uk
http://www.cs.man.ac.uk/~franconi/

Abstract

In this paper we present i•com, a tool for intelligent conceptual modelling. i•com allows for the specification of multiple EER diagrams and inter- and intra-schema constraints. Complete logical reasoning is employed by the tool to verify the specification, infer implicit facts, and manifest any inconsistencies.

1 Introduction

i•com is a tool supporting the conceptual design phase of an information system, and in particular of an integration information system – such as a data warehouse. The tool is an evolution of part of the conceptual modelling demonstrators suite [Jarke *et al.*, 2000] developed within the European ESPRIT Long Term Research *Data Warehouse Quality* (DWQ) project [Jarke *et al.*, 1999]. i•com adopts an extended Entity-Relationship (EER) conceptual data model, enriched with multidimensional aggregations and interschema constraints. i•com is fully integrated with a very powerful description logics reasoning server which acts as a background inference engine.

The conceptual modelling language supported by i•com can express:

- the standard Entity-Relationship data model, enriched with IsA links, disjoint and covering constraints, full cardinalities, and *definitions* attached to entities and relations by means of view expressions over other entities and relationships in the schema [Calvanese *et al.*, 1998c];
- aggregated entities together with their multiply hierarchically organised dimensions [Franconi *et al.*, 1999, Franconi and Sattler, 1999];
- (interschema) constraints, such as inclusion and equivalence between expressions involving entities and relationships possibly belonging to different schemas [Catarci and Lenzerini, 1993, Calvanese *et al.*, 1998b].

The tool supports multiple schemas with interschema constraints but it turned out to be extremely useful also in supporting the conceptual modelling of "classical" databases involving a single rich schema with integrity constraints, and in designing ontologies for various purposes.

i•com reasons with (multiple) diagrams by encoding them in a single description logic knowledge base, and shows the result of any deductions such as inferred links and inconsistent entities or relationships. Theoretical results from the DWQ project guarantee the correctness and the completeness of the reasoning process: the system uses the $\mathcal{SHIQ}$ description logic [Horrocks *et al.*, 1999], which provides the core expressivity of the $\mathcal{DLR}$ logic developed by [Calvanese *et al.*, 1998a]. To the best of our knowledge, this is the first implemented tool for EER conceptual modelling with a provably *complete* inference mechanism for consistency checking and for deduction – i.e., derivation of implied links in the schema. Completeness of reasoning means in this context that no valid deduction is left out by the inference engine. This of course holds for the full data model employed by i•com, which is much richer than EER.

As exemplified by the scenario proposed in Section 3, i•com fully supports the conceptual design methodology studied within the DWQ project [Calvanese *et al.*, 1998b, Calvanese *et al.*, 1998a], where the consistency and deduction reasoning tasks play a crucial role.

The tool allows for the creation, the editing, the managing, and the storing of several interconnected conceptual schemas, with a user friendly graphical interface (including an auto-layout facility). The i•com tool (GUI client + i•com server) is written in standard Java 1.2, and it is being used on Unix and Windows machines. i•com communicates via a CORBA protocol with the FaCT description logic server [Horrocks, 1999]. Experiments with i•com show that it is able to handle the large *integrated Conceptual Data Warehouse Model* of a national European telecom company. i•com provides an interface for importing and exporting Rational Rose™ diagrams.

2 The Modelling Language

For the modelling of a single schema, an extended Entity-Relationship (EER) conceptual data model has been adopted. Basic elements of ER schemas are *entities*, denoting a set of objects called *instances*, and *relationships*, denoting a set of *tuples* made by the instances of the different entities involved in the relationship. Since the same entity can be involved in the same relationship more than once, participation of entities in relationships is represented by means of *ER-roles*, to which a unique name is assigned. ER-roles can have *cardinality constraints* to limit the number instances of an entity involved in the relationship. Both entities and relationships can have *attributes*, i.e., properties whose value belong to some predefined domain – e.g., `Integer`, `String`. Additionally, the EER model includes *taxonomic* relations to state inclusion assertions between entities and between relationships, with the possibility to specify optional *covering* and *disjointness* constraints.

The most interesting feature of the extended modelling language is the ability to com-

$$
\begin{array}{lcl}
R & \rightarrow & \top_n \mid RN \mid \neg R \mid R_1 \sqcap R_2 \mid i/n : E \\
E & \rightarrow & \top_1 \mid EN \mid \neg E \mid E_1 \sqcap E_2 \mid \exists^{\geq k}[i]R
\end{array}
$$

$$
\begin{array}{rcl}
(\top_n)^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n \\
RN^{\mathcal{I}} & \subseteq & (\top_n)^{\mathcal{I}} \\
(\neg R)^{\mathcal{I}} & = & (\top_n)^{\mathcal{I}} \setminus R^{\mathcal{I}} \\
(R_1 \sqcap R_2)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} \\
(i/n : E)^{\mathcal{I}} & = & \{\langle d_1 \ldots d_n \rangle \in (\top_n)^{\mathcal{I}} \mid d_i \in E^{\mathcal{I}}\}
\end{array}
\qquad
\begin{array}{rcl}
(\top_1)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
EN^{\mathcal{I}} & \subseteq & (\top_1)^{\mathcal{I}} \\
(\neg E)^{\mathcal{I}} & = & (\top_1)^{\mathcal{I}} \setminus E^{\mathcal{I}} \\
(E_1 \sqcap E_2)^{\mathcal{I}} & = & E_1^{\mathcal{I}} \cap E_2^{\mathcal{I}} \\
(\exists^{\geq k}[i]R)^{\mathcal{I}} & = & \{ d \in (\top_1)^{\mathcal{I}} \mid \\
 & & \quad \sharp\{\langle d_1 \ldots d_n \rangle \in R^{\mathcal{I}} \mid d_i = d\} \geq k\}
\end{array}
$$

Figure 1: $\mathcal{DLR}$ and its semantics.

pletely *define* entities and relationships as *views* over other entities and relationships of the conceptual schema [Calvanese *et al.*, 1998c]. The adopted view language is $\mathcal{DLR}$, a description logics over unary and *n*-ary relationships. The basic types of the $\mathcal{DLR}$ are *entities* (also called *concepts*) and *n-ary relationships*. According to the syntax rules at the top of Figure 1, $\mathcal{DLR}$ entity expressions (denoted by E) are built out of *primitive entities* (denoted by EN) and relationship expressions (denoted by R) are built out of *primitive relationships* (denoted by RN). Informally, the *semantics* of an entity expression is, as expected, a set of objects, and the meaning of n-ary relationships is a set of n-tuples of objects. Both for entity and relationship expressions, the full boolean calculus is allowed. The expression $(i/n : E)$ denotes a n-ary relationship whose ith argument is selected to be of type E. The expression $(\exists^{\geq k}[i]R)$ denotes the projection over th ith argument of the relationship R, for only those tuples whose cardinality of the ith argument is at most k. The equations at the bottom of Figure 1 formally define the semantics of $\mathcal{DLR}$.

$\mathcal{DLR}$ is an interesting decidable fragment of first order logic: among others, inclusion dependencies with $\mathcal{DLR}$ views can express the following classes of constraints:

- key dependencies;
- typed inclusion dependencies without projection;
- existence dependencies;
- exclusion dependencies.

When dealing with multiple schemas modelling different information sources, i•com may express interschema knowledge by means of $\mathcal{DLR}$ inclusion dependencies. These are used to state the interdependencies that hold between entity and relationship expressions in the different conceptual schemas, and have been introduced as *intermodel assertions* in [Catarci and Lenzerini, 1993]. We assume that each domain, entity, relationship, attribute, and ER-role symbol A of each schema $\mathcal{S}_i$ is identified by a unique symbol A_i; intermodel assertions are thus dependencies between views involving symbols of the different schemas.

Calls (av. duration in secs.)		Destination Area Code							
		0387	0338	0355	06	0461	···	···	···
Source (Point Type)	Land Line	34	123	56	537	230	···	···	···
	Cell	60	34	69	43	212	···	···	···
	Direct Data	507	342	360	456	231	···	···	···
	PABX	123	213	192	336	271	···	···	···

Figure 2: A multidimensional cube of *telephone calls* showing their average duration.

It is worth noting that the integration process is incremental – since the resulting *integrated* schema can be monotonically refined as soon as there is new understanding of the different component schemas – and that the resulting unified schema is strongly dependent from (actually, it includes) the single information sources.

In the particular but important case of designing a Data Warehouse Conceptual Schema it is assumed to have a privileged schema – the *global* view – which is the conceptual representation of the global concepts and relationships reconciled and abstracted in the data warehouse, and it is not necessarily a complete model of all the source information. Such schema is integrated with the different source schemas. An important point is that not only the interrelationships between the source schemas and the global schema are modelled, but also the interdependencies between the source schemas themselves. Moreover, the global integrated schema – the Data Warehouse Conceptual Schema – is composed not only by the global schema, but also by the various source schemas and by the intermodel assertions.

An extension of the EER formalism for conceptual modelling multidimensional aggregations has been included in i•com. The extended conceptual data model is able to represent the structure of *aggregated entities* and of *multiply hierarchically organised dimensions* [Franconi and Sattler, 1999]. The idea is to allow for the abstract conceptual representation of multidimensional cubes – such as the one of Figure 2 – as full fledged entities in the EER diagram. Aggregations become entities in the EER schema, and thus they are first class citizens of the representation language. It is possible to describe the components of aggregations, and the relationships that the properties of the components may have with the properties of the aggregation itself; it is possible to build aggregations out of other aggregations, i.e., it is possible for an aggregation to be explicitly built on top of aggregated dimensions. A multidimensional modelling object in the logical perspective—e.g., a materialised view in the star or snowflake model, a query, or a (multidimensional) cube—should always be related with some (possibly aggregated) entity in the conceptual schema specified in i•com. For a complete description of the data model refer to [Franconi and Sattler, 1999].

A limit of the adopted formalism is that the description of an aggregation may not include a specification of *how* values of its attributes are computed from attribute values of its components using aggregation functions such as MIN, AVERAGE, or SUM. The reason for this comes from an important result of the research within the DWQ project which identifies the borders for the possible extensions of a Data Warehouse Conceptual Data Model towards the explicit inclusion of aggregation functions [Baader and Sattler, 1998].

3 A Modelling Scenario

As we have noticed before, conceptual modelling is a highly iterative process. We assume that we start with a (possible empty) partial schema and want to add some new information, that is, some additional objects or properties of objects that a user is interested in and that thus should be modelled conceptually. The tool provides support for the design of the schema by making explicit all (implicit) consequences of facts that have been modelled so far. Moreover, it will also detect all inconsistencies in the schema. Likewise, the new schema that is the output of the modelling process does not only contain all the facts we have explicitly modelled, but also the implicit consequences that were detected by the reasoner together with the freshly measured quality factors of the objects in the new schema. This supports the designer during the conceptual modelling process in understanding whether the schema developed so far actually captures the intended meaning. This either increases the belief in the correctness of the conceptual schema, or gives an argument for its inconsistency.

3.1 Conceptual Modelling Source Integration

Let us consider as an example the conceptual design of a data warehouse in telecommunication business. In this warehouse, we want to integrate data from a source that contains information concerning specific telephone calls. Let us suppose that the source database contains a table relating the phone calls done by the customers of the company with the locations of the calls themselves. The schema is ambiguous in the sense that it is not clear whether the location associated to each call indicates the source or the destination of the call. In the attempt to reuse this data in a new centralised system collecting information for analysis about mobile phones, the source schema is related to a global integrated schema talking about mobile calls, where the origin and the destination of the calls are differentiated (figure 3). The `Service` entity of the source schema is then asserted to generalise the `MobileCall` entity in the global schema and the `ServicePlace` entity of the source schema generalises the `PhonePoint` entity in the global schema. The lack of information about the real meaning of the `service-location` entity in the source schema is captured by the designer with the assertion that such a relation generalises both the `destination` and the `source` relationships in the global schema. A closer took at the source database, during a reverse engineering phase, reveals to the designer an additional interschema constraint stating that whenever we consider `Cell` points as `ServicePlace` in the source table, then the meaning of the allegedly underspecified relationship is to denote the `destination` of the call. Thus, the view `service-cell` is defined as a `service-location` relationship restricted to consider `Cell` points only, and it is asserted to be a specialisation of the `destination` relationship.

At this point, the following fact is logically implied by the integrated schema: for every call, any source cell point is also a destination point. This fact is clearly a symptom that something went wrong. Indeed, the designer can not accept this conclusion. In this example,

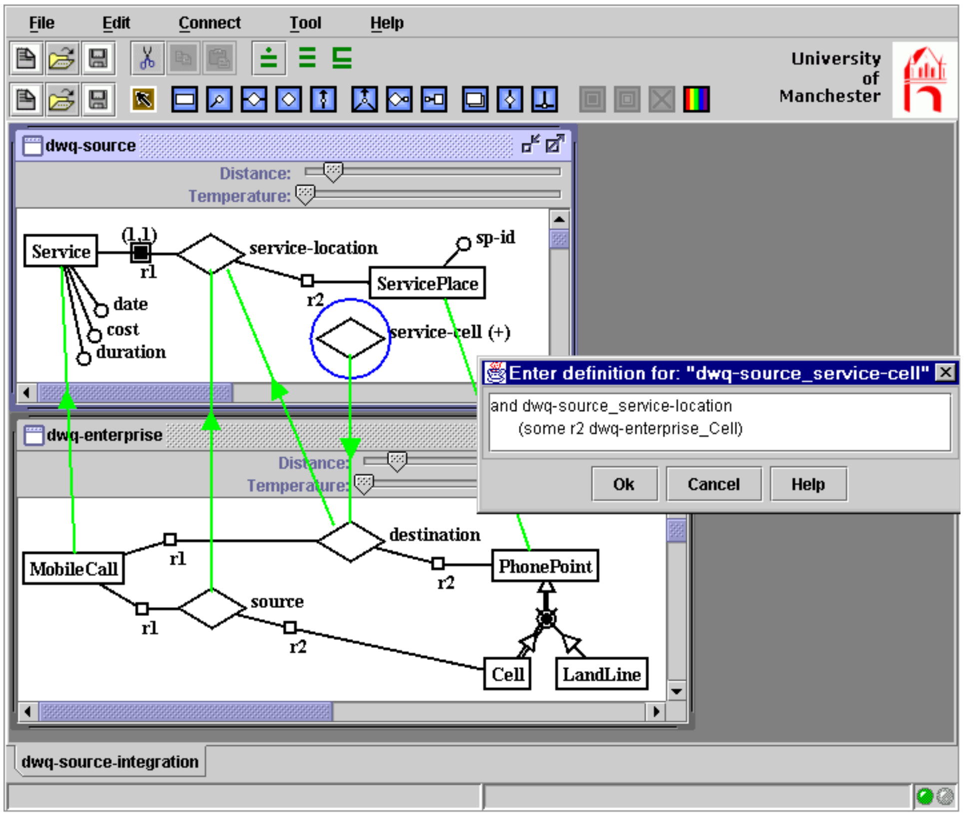

Figure 3: Schema Integration with i•com.

the reason for this can be found in the wrong assumption that the original relationship was modelling both sources and destinations. If we omit in the integrated schema that the source relationship specialises the `service-location` relationship, then the above deduction does not hold anymore, and the model is acceptable.

3.2 Conceptual Modelling Multidimensional Aggregation

The conceptual modelling of the client schema differs from the overall conceptual modelling in being oriented to the specific information needs of the different clients/users. Depending on the clients area of responsibility and the task to be addressed, clients are not only interested in specific portions of the data warehouse but also in specific forms of representation regarding both the granularity and the multidimensional aggregations used to derive high-level information. The conceptual client schema aims at capturing these specific client demands.

In order to optimally satisfy the clients information needs, the treatment of user-specific multidimensional aggregations on a conceptual level is indispensable. On the one hand, the conceptual modelling of aggregations supports the specification of adequate quality factors for aggregations on the conceptual level and thus allows to take into account quality factors for multidimensional cubes (both for MOLAP and ROLAP data models) already during the

Mobile Calls (av. duration in secs.)		Destination Area Code							
		0387	0338	0355	06	0461	···	···	···
Source (Point Type)	Land Line	X	X	X	X	X	X	X	X
	Cell	60	34	69	43	212	···	···	···
	Direct Data	507	342	360	456	231	···	···	···
	PABX	123	213	192	336	271	···	···	···

Figure 4: A multidimensional cube of *mobile telephone calls* showing their average duration.

data warehouse design process [Franconi *et al.*, 1999]. On the other hand, the conceptual schema of multidimensional aggregation can be exploited for semantic query optimisation.

Coming back to our telecom example, a useful aggregation for analysing the nature of telephone calls may consider, among others, the dimension related to the origin of the calls (land line, cell, etc.) – in order to compute, say, their average duration (see Figure 2). In the context of this particular data warehouse, such an aggregation makes sense, but if just *mobile* calls are considered, then the corresponding cube will show now an "empty" line

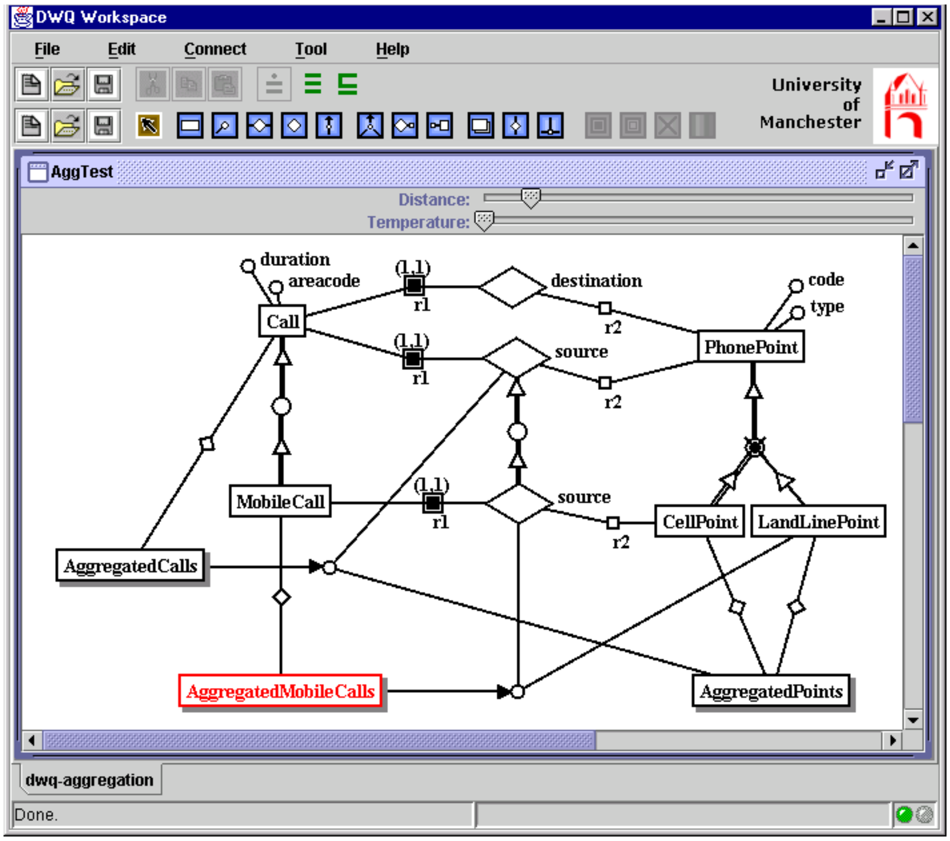

Figure 5: Modelling Aggregations with i•com.

(i.e., filled with null values) corresponding to land-line type mobile calls (see Figure 4): in fact, in this case no aggregated entity may exist of mobile calls originating from non cell points. Thus, if in the global schema such a refined aggregation considering mobile type calls is introduced, the reasoner would tell immediately that such entity is inconsistent. This means that no materialised views may exist in the data warehouse related to this aggregation. Using the extended Entity Relationship formalism supported by i•com, this inconsistency would be detected. This is shown in figure 5, where the entity `AggregatedMobileCalls` aggregating `MobileCalls` according to the `source` dimension at the `LandLine` point level is derived to be inconsistent, while the `AggregatedCalls` entity is consistent since it could be filled by the values concerning mobile calls.

Acknowledgements

We wish to thank the DWQ team, and in special way Diego Calvanese and Maurizio Lenzerini from Rome, and Uli Sattler and Franz Baader from Aachen, who contributed to provide most of the ideas and the theoretical grounds exploited by i•com. Nicola Pedot, supervised by Paolo Bresciani, implemented a first version in 1998; Christophe Sanchez developed the initial modular translator.

References

[Baader and Sattler, 1998] Franz Baader and Ulrike Sattler. Description logics with concrete domains and aggregation. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 336–340, 1998.

[Calvanese *et al.*, 1998a] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13. Morgan Kaufmann, 1998.

[Calvanese *et al.*, 1998b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[Calvanese *et al.*, 1998c] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*. Kluwer, 1998.

[Catarci and Lenzerini, 1993] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[Franconi and Sattler, 1999] Enrico Franconi and Ulrike Sattler. A data warehouse conceptual data model for multidimensional aggregation. In *Proceedings of the Workshop on Design and Management of Data Warehouses (DMDW'99)*, 1999.

[Franconi *et al.*, 1999] Enrico Franconi, Franz Baader, Ulrike Sattler, and Panos Vassiliadis. Multidimensional data models and aggregation. In Jarke et al. [1999], chapter 5, pages 87–106.

[Horrocks *et al.*, 1999] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[Horrocks, 1999] Ian Horrocks. FaCT and iFaCT. In *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

[Jarke *et al.*, 1999] Mathias Jarke, Maurizio Lenzerini, Yannis Vassilious, and Panos Vassiliadis, editors. *Fundamentals of Data Warehousing*. Springer-Verlag, 1999.

[Jarke *et al.*, 2000] Matthias Jarke, Christopg Quix, Diego Calvanese, Maurizio Lenzerini, Enrico Franconi, Spyros Ligoudistiano, Panos Vassiliadis, and Yannis Vassiliou. Concept based design of data warehouses: The DWQ demonstrators. In *2000 ACM SIGMOD Intl. Conference on Management of Data*, 2000.