The Conceptual Integration Modelling Framework: Semantics and Query Answering

by

Ekaterina Guseva

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
for the degree of Master in Computer Science

Ottawa-Carleton Institute for Computer Science
Faculty of Graduate and Postdoctoral Studies
University of Ottawa

Acknowledgements

First, I would like to thank my supervisor, Dr. Iluju Kiringa, for giving me the opportunity to do my research and for his significant effort in helping me. I would like to express respect for his patience, motivation and guidance.

Finally, I would like to express my appreciation to my mother , whom I owe everything I have; I am here now only because of her. She has always been a great example of strength, determination and devotion to her family.

Abstract

In the context of business intelligence (BI) the accuracy and accessibility of information consolidation play an important role. Integrating data from different sources involves its transformation according to constraints expressed in an appropriate language. The conceptual integration modelling framework (CIM) acts as such a language. The CIM is aimed to allow business users to specify what information is needed in a simplified and comprehensive language. Achieving this requires raising the level of abstraction to the conceptual level, so that users are able to pose queries expressed in a conceptual query language (CQL).

The CIM is comprised of three facets: an Extended Entity Relationship (EER) model (a high level conceptual model that is used to design databases), a conceptual schema against which users pose their queries, a relational multidimensional model that represents data sources, and mappings between the conceptual schema and sources. Such mappings can be specified in two ways: in the first scenario, the so-called global-as-view (GAV), the global schema is mapped to views over the relational sources by specifying how to obtain tuples of the global relation from tuples in the sources. In the second scenario, sources may contain less detailed information (a more aggregated data) so the local relations are defined as views over global relations that is called as local-as-view (LAV).

In this thesis, we address the problem of expressibility and decidability of queries written in CQL. We first define the semantics of the CIM by translating the conceptual model so we could translate it into a set of first order sentences containing a class

of conceptual dependencies (CDs) - tuple-generating dependencies (TGDs) and equality generating dependencies (EGDs), in addition to certain (first order) restrictions to express multidimensionality. Here a multidimensionality means that facts in a data warehouse can be described from different perspectives. The EGDs set the equality between tuples and the TGDs set the rule that two instances are in a subtype association (more precise definitions are given further in the thesis).

We use a non-conflicting class of conceptual dependencies that guarantees a query's decidability. The non-conflicting dependencies avoid an interaction between TGDs and EGDs. Our semantics extend the existing semantics defined for extended entity relationship models to the notions of fact, dimension category, dimensional hierarchy and dimension attributes.

In addition, a class of conceptual queries will be defined and proven to be decidable. A DL-Lite logic has been extensively used for query rewriting as it allows us to reduce the complexity of the query answering to AC_0. Moreover, we present a query rewriting algorithm for the class of defined conceptual dependencies.

Finally, we consider the problem in light of GAV and LAV approaches and prove the query answering complexities. The query answering problem becomes *decidable* if we add certain constraints to a well-known set of EGDs+TGDs dependencies to guarantee summarizability. The query answering problem in light of the global-as-a-view approach of mapping has AC_0 data complexity and *EXPTIME* combined complexity. This problem becomes *coNP* hard if we are to consider it a LAV approach of mapping.

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Nowadays, organizations are faced with the issue of exponential data growth. Over the last two decades, data warehouses have served as a paramount tool for storing the gigantic amounts of (largely relational) data that is accumulated by organizations. A data warehouse is a data repository that gathers information about the world facts from different perspectives. Data warehouses have a popular representation in the form of star or snowflake schemas [48] that express multidimensional facts.

A multidimensional fact characterizes world facts from different perspectives. These perspectives are summarized in hierarchical dimensions, which can describe a world fact at different levels of generality.

Usually, the notions of a fact table (table that stores data about world facts) and a dimension table (that stores information about perspectives a fact is described from) stands at the center of these schemas.

This thesis falls within the context of business intelligence (BI). The latter is a software industry term that refers to the use of information within organizations to allow decision-makers, usually executives, to make informed decisions, and to effectively run operations using the known data which is stored in a data warehouse typically. As a

branch of Computer Science, it encompasses several areas, among which data integration and warehousing plays a central role. Data integration deals with the information consolidation coming from multiple sources into a unified repository. Ideally, integration should happen almost automatically without any human intervention. One important aim of BI is to be user oriented: decision makers and business users need to focus only on modelling their data needs in (usually high-level) business terms (conceptual schemas) that they fully understand. These needs can be expressed in the conceptual integration model (CIM) language [62]. The aforementioned data warehouse (star and snowflake) schemas can be represented using the extended entity relationship model notions augmented with certain restrictions. The conceptual integration modelling framework is a top-down data modelling approach in which business analysts are required to have all the information about the data stored in the (relational) data warehouse sources so the analyst will only create conceptual schemas tailored to their own needs.

The recent work has focused on designing [57] and implementing the CIM model [33]. A business user would specify what information is needed and in what form, and the integration system would satisfy the request by producing the integrated, usually multidimensional data as automatically as possible. To achieve this, one must raise the level of abstraction significantly from the data level towards the conceptual level to accommodate business users.

Technically, the CIM extends Microsoft's Entity Data Model (EDM) (See the literature review later in this thesis) with multidimensional modelling concepts [57]. The EDM framework has three components, namely a conceptual level component made of an extended entity relationship model, a relational data store level made of relational schemas, and a mapping between elements of the conceptual and data level schemas. Likewise, the CIM has three components: the conceptual visual language (CVL) which is a multidimensional conceptual schema language for representing the highest level of abstraction, the storage visual language (SVL) which is a multidimensional schema language (which is the usual data warehouse schema), and the visual mapping language (VML)

which is a mapping language. The latter is used to specify correspondences between conceptual schema elements and multidimensional schema elements. User mappings are subsequently compiled into a set of views over the underlying multidimensional schemas. A query processing in this setting amounts to answering queries using views [33].

1.2 The Problem

Conceptual level schemas as introduced by the EDM framework are essentially used as run-time environments. As indicated earlier, both the EDM framework and the CIM framework offer a run-time environment in which the conceptual schemas are compiled into views that are subsequently used to answer queries at run-time. These frameworks have been developed to offer an application whose formal foundations need to be posed. That is, essential questions pertaining to the semantics of the framework are required to be answered such as: What are the semantics of the proposed schemas, and what can be expressed in these schemas? Further, given a practical query formulated in an appropriate query language, what is the complexity of answering the query when posed against the considered conceptual schema? With respect to the EDM framework and any similar framework where EER model schemas are mapped to underlying relational schemas, formal foundations have been given in the works described in [20, 25]. Here the authors define mappings between a (non-multidimensional) EER conceptual schema and a relational schema, and present the semantics of these schemas as well as the computational complexity of processing appropriate queries against them.

To define complexity, we consider two cases. In the first case, we consider the complexity of evaluating a conjunctive query on a relational database, where both the query and the database are not fixed and have its influence on the query answering. The complexity of this problem is usually referred to as combined complexity. On the other hand, the complexity of the problem of evaluating a query on a relational database, where the query is assumed to be fixed and the data is not fixed, is called data complexity.

In this thesis, we will address the problem of ascribing formal semantics to the CIM schemas. We will also address the problem of the complexity of answering conceptual queries posed against CVL schemas, as well as the expressiveness of these schemas. Moreover, we will study the problem of capturing the fundamental property of summarizability that is associated with multidimensional hierarchical schemas. By definition, a dimension is summarizable, if aggregations at finer levels in a multidimensional hierarchy can be used to compute aggregations at higher levels.

For example, consider the location hierarchy given by Hurtado et al. [43]

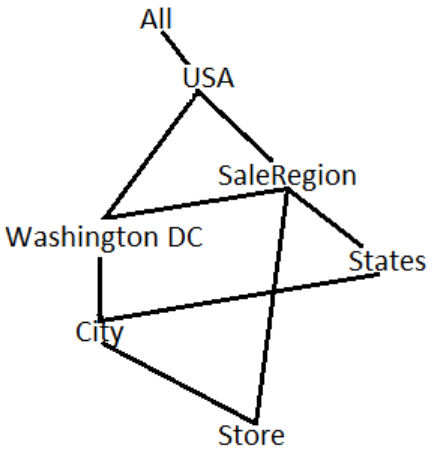


Figure 1.1: The location dimension

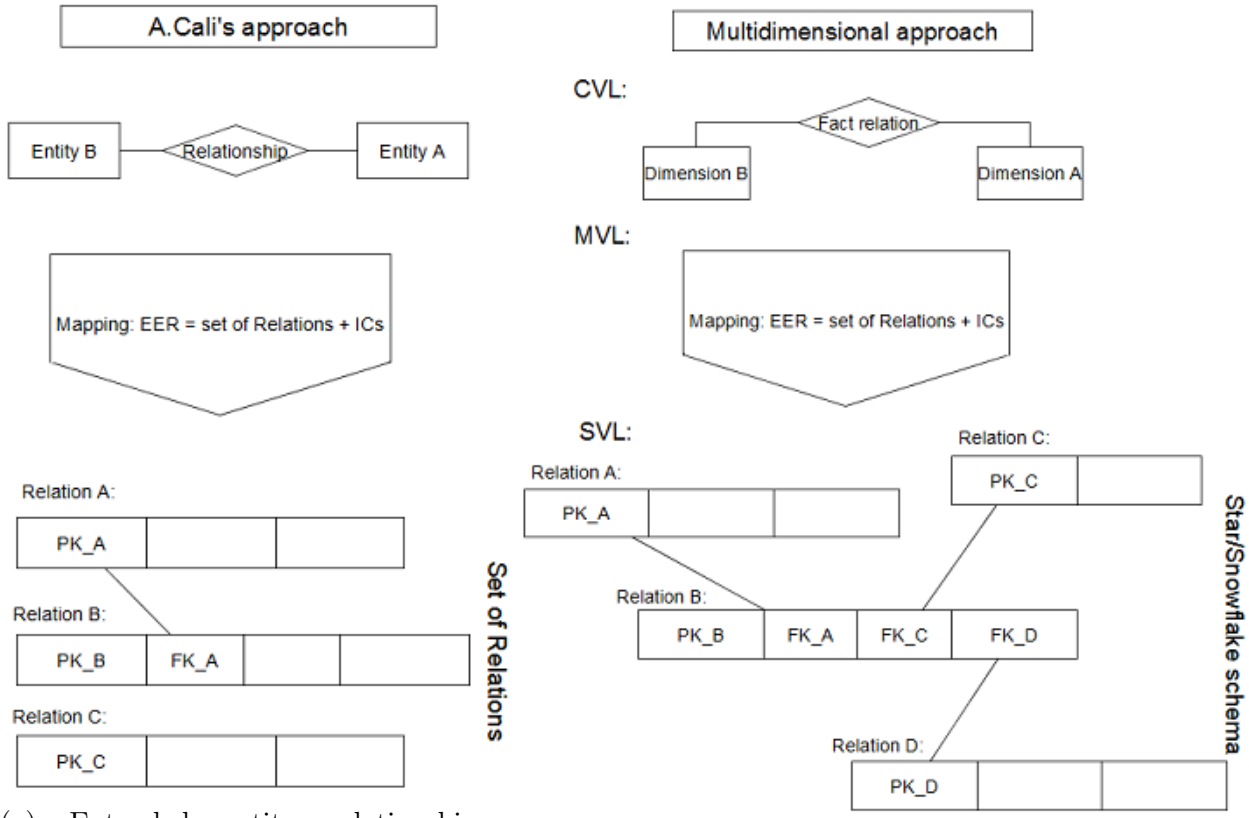
The United States contains states, and Washington DC which is a district and does not belong to any of the states. Consequently, we cannot compute the view for the country (USA) from the view for Washington DC. Thus, this is an example of unsummarizable dimension.

We will enrich the existing semantics studied in [20] to address summarizability. In fact, we will show that constraints presented in [20, 25] are not enough to capture the so-called summarizability of dimensions in a star/snowflake schemas. Summarizability guarantees the correctness of views prestored in the data warehouse. It is highly important to assure that results obtained by roll-up operations along the existing dimensions are correct.

A roll-up is one of the data warehouse operations that performs aggregation by sum-

marizing the data at a finer level by generalizing the result. The summarizability requires additional constraints to apply to the conceptual schema in order to perform correct queries. Studying query answering will be done by rewriting user queries using DL-Lite ontology [25]. We employ DL ontology because it is a logical query language for checking satisfiability of a knowledge base and answering complex queries. It has the advantage of low cardinality and better expressiveness. Also a vast class of queries over DL-Lite knowledge base are first-order rewritable.

1.3 The Approach



(a) Extended entity relationship framework

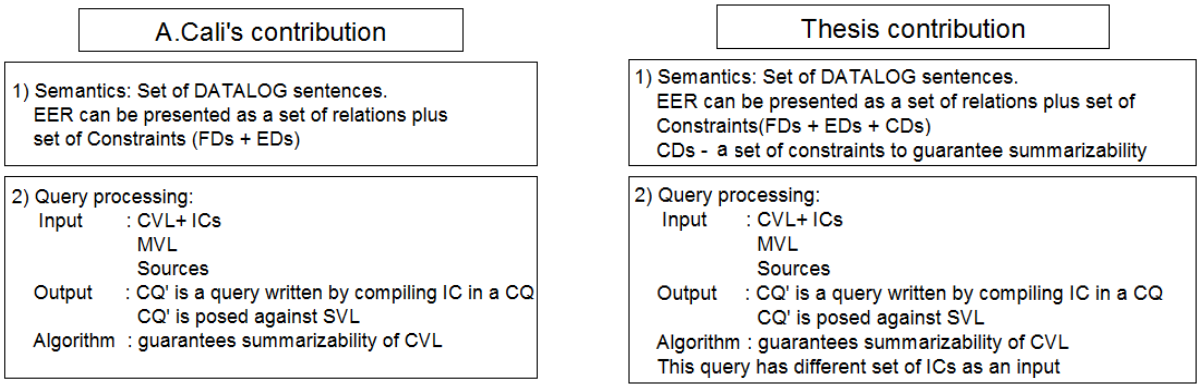
(b) CIM for multidimensional schema

Figure 1.2: CIM approach comparison

In [20], the authors formulated an approach for data integration, where the conceptual

schemas were translated to a relational model augmented with inclusion dependencies and key dependencies. A graphical representation of this framework is illustrated in Figure 1.2a.

Here we can see that the conceptual schema that Cali et al. [20] has defined is represented using entities and relationships between entities. As it can be seen the proposed approach is quite similar to the one that is taken in this thesis. In both approaches the semantics for the global conceptual schemas is defined as a set of Datalog sentences and set of conceptual dependencies. Differently from [20], we enrich the set of dependencies to capture multidimensionality. However, Cali's approach does not capture the notion of fact relationship between hierarchical dimensions. In our case, the relational sources have either star or snowflake schema (SVL). Furthermore, the relationship between dimensions and the fact relation is represented by the fact relation's compound primary key. Our approach is shown in the Figure 1.2b. We can present our CIM schemas as a sets of relations augmented with conceptual dependencies similar to the approach in [20]. However, in our case we will have to add certain constraints to ensure summarizability.



(a) EER Framework approach

(b) CIM approach

Figure 1.3: CIM approach comparison

Once the semantics are defined we can address the problem of query answering. We present the algorithm to compile the conceptual dependencies in a query so it can be posed against sources directly. The approach is presented in a Figure 1.3. Similarly to

the approach in [25], as an input for the algorithm, we have a global conceptual schema with the set of conceptual dependencies. Although, in our case we have a different set of constraints to guarantee summarizability of the hierarchical dimensions of the global schema. We will have to perform certain operations for the query to find dimensions that are used and to build a hierarchical graph to test the summarizability. In the algorithm we will have to process each atom of the query and build its dimensional hierarchy to check summarizability. As an output of the algorithm we have a rewritten query with the imposed constraints. This query can be posed against multidimensional sources.

The mappings between CVL schemas (as global schemas) and the SVL schemas come in two varieties, namely the GAV and LAV approaches. In the GAV approach, the global schema is mapped to source schema elements contained in the SVL schema. On the other hand, in the LAV approach, the SVL schema elements are considered to be views over the elements of the CVL schema. This thesis studies the complexity of answering certain classes of queries for both cases.

We tailor an algorithm defined in [25] for transforming user queries into a set of conjuncts imposing inclusion and functional constraints to our needs. We prove the data and combined complexity of the query answering using this adapted algorithm.

1.4 Contributions

This thesis makes the following contributions:

1. **Semantics of Multidimensional Conceptual Models**: Inspired by the work in [18], we show the similarities of both approaches. We apply the tuple-generating dependencies (TGDs) and the equality-generating dependencies (EGDs) used for EER schemas in [18] to CVL schema and to mappings of the latter to relational multidimensional sources. Doing so, we show that the CIM schemas require more expressive semantics than the usual EERM schemas in order to capture summarizability for the dimensions. Thus, we augment the semantics defined in [18] with

additional syntactic notions (e.g. categories) and certain constraints.

2. **Decidability of Results**: Since the semantics of the CVL is defined via generating views (prestored queries) over the underlying multidimensional data stores, we prove decidability results with respect to the generated views augmented with appropriate constraints.

3. **Complexity of Conceptual Query Answering**: We design a query processing algorithm that compiles the multidimensional constraints into the query that is subsequently posed against the CVL schema. We show that the algorithm has AC_0 data complexity and *EXPTIME* combined complexity.

4. **Complexity of GAV and LAV Conceptual Query Answering**: Finally, we prove complexity results by taking into account the GAV and LAV approaches of mapping.

1.5 Thesis Outline

The thesis is organized as follows: Chapter 2 provides a literature review as well as the background work done on the CIM framework as well as on multidimensional databases. In Chapter 3, we present design details of the CIM framework. We describe the main components of the framework language by means of an example. The chapter then shows how mappings are performed on one of the dimensions of the given example. We also describe the semantics of the CVL schemas and give a detailed illustration of it using a sales data warehouse example. In Chapter 4, we describe the proposed algorithm for query processing and prove several data and combined complexity results. Results are provided for GAV and LAV approaches. Finally, we reach several conclusions along with a brief overview of future work.

Chapter 2

Related Work

In this chapter we give an overview of the related work. The related topics have been widely studied by many authors, however, there are some gaps to be filled. The complete chapter has been split into several sections according to topics that are related to our problem addressed in this thesis. Those sections are: conception integration model, here we give a short overview of the existing data integration frameworks; multidimensional database constraints, where we summarize the work that has been done in order to define constraints that can capture hierarchical multidimensionality; data source integration and query processing, here we give an overview of existing algorithms for query processing; in the section regarding data warehouse semantics, we describe the existing semantics for star/snowflake schemas.

2.1 Datalog Framework for query answering

A Datalog is an ontology language that syntactically is a subset of Prolog. It is often used as a query language for deductive databases [72]. The Datalog is widely used in data integration and information extraction. Datalog language is expressive enough to use it for formulating our semantics and queries. Moreover, as we will describe later a special class of Datalog benefits from low complexity.

A Datalog program [2] defines the relations that occur in the head of rules based on other relations. The definition is recursive, so the defined relations can also occur in bodies of rules. Thus, a Datalog program is interpreted as a mapping from instances over the relations occurring in the bodies only, to instances over the relations occurring in the heads.

A Datalog rule is an expression of the form $head \leftarrow body$, where the head is a relational atom and the body contains one or more atoms called subgoals. The head is true for the variables if there exist values that make all subgoals of the body true.

For example, $Employee(Employee\_id) \leftarrow Sales\_Represent(Employee\_id)$.

This example means that each sales representative is an employee, only in this case the body $Employee(Employee\_id)$ returns true.

A Description logic is a formal knowledge representation language designed to assist the formal description of databases. In the Description logic there are two types of states - $ABox$ and $TBox$

An ABox [71] is an "assertion component" of the knowledge base which means it formulates statements. For example,
$John \rightarrow Sales\_Represent$.

TBox [73] is a "terminological component" — a conceptualization associated with a set of facts. For example,
$Product(Product\_id; Category\_id) \leftarrow Category(Category\_id)$,

2.2 Conceptual Integration Model

The conceptual integration modelling (CIM) project is a data warehouse research project that has been conducted at the University of British Columbia and the University of Ottawa. This project addresses the problem of mapping a conceptual level schema to a multidimensional data level that represents information that has already been integrated

from data sources. The project aims to create a framework for mapping a data warehouse schema to a global multidimensional conceptual schema. At the time of design, a data warehouse schema is constructed based on the local schemas of the data sources to be integrated. The designed data warehouse schema is subsequently mapped to a higher level multidimensional conceptual schema. At run time, the warehousing process compiles the given mappings into views over the data warehouse schema. The produced views are then used for posing conceptual schema queries against global conceptual schema [62].

The conceptual modeling language in the CIM has two layers: (1) the CIM visual model (CVM) and (2) the CIM data model (CDM). Here, we focus on the CVM; specifically, on the formal foundations of the conceptual visual language (CVL) which is a visual representation of the multidimensional conceptual schema, and on the conceptual mapping language used to map conceptual schema elements to data warehouse schema elements. CVL is inspired by MultiDim [34] and StarER [68].

In [65] the authors defined a formal semantics of the conceptual modeling language by giving a precise definition for the syntax so in the future it could be mapped to a storage visual language. In this thesis, a fully functioning graphical editor was developed for the model manager to draw a conceptual model in a drag - and - drop fashion.

In the work that follows, we will concentrate on the next step - defining a mapping visual language. The CIM allows an expression of mappings between the conceptual schema and the schemas of the data sources [57]. The CIM model uses mappings of the common form $c \rightsquigarrow \psi_S$, where c is an element in the target schema and ψ_S is a query (or view) over the source schema (in the GAV approach of mapping).

These internal languages such as mapping visual language allow ad - hoc manipulation. We will refer an enthusiastic reader to [62]. In this thesis, we only focus on the MVL.

As an extension of the proposed CIM in [12] the authors propose a business intelligence model that helps express user needs and perform the analysis of the data collected by the enterprise more effectively. This model works together with the CIM with the

latter providing access to multidimensional data through mapping layer. The authors expect that at design time, a business analyst would specify in the CIM what information is needed and in what form, so that the system could respond to business user queries in terms of business intelligence model (BIM) concepts at run - time. The work of [11] gives an integrated framework overview and present the connection between BIM and the CIM in particular.

The research project described in [14] was conducted by the intelligence and information section at Defense Research and Development Canada (DRDC) to create a platform for scalable multi - intelligence data integration services (MIDIS) in order to support flexible data integration approach from different heterogeneous sources that use Semantic Web and Big Data technologies. The heterogeneous sources can be any source of information including images, report documents and any other type of unstructured data. The paper represents a very high - level architecture of the platform without giving any semantic base for its integration. According to [69] the query answering complexity in this case could reach $NEXPTIME$. The key components include: data extraction from heterogeneous sources; ontology - based semantic enrichment;data querying and analytic.

In [75] the authors describe the process of data source integration and listed existing integration solutions that have been proposed by the database community. They gave an overview of different approaches to address the integration problem and included further details regarding some of them. Interestingly, the paper describes different approaches based on the level of abstraction where integration is performed. In the following we will list some of the integration solutions described in this paper.

In [7], Services and Information Management for decision Systems (SIMS), similar to our CIM, is aimed to help users formulate their queries in a form that is more familiar to them in order to avoid them dealing with unnecessary details of data distribution. A global conceptual schema is used as a single ontology and mapping language has been defined to relate data sources to it; queries are then posed against this global schema. Similar to the CIM, SIMS proceeds to rewrite the query to a set of statements that refer

to data sources. In our CIM model we follow a different aim to perform the rewriting algorithm, namely, to impose constraints induced by the global schema. SIMS then creates a plan to retrieve the information via performing corresponding subqueries. Similar to SQL query optimizer, using the information about data sources to be integrated, SIMS reformulates the plan to minimize execution time.

The goal of the Stanford - IBM manager of multiple information sources (TSIMMIS) [29] is to provide tools for data integration from multiple information sources, and to ensure that the integrated data is consistent. According to its architecture, the wrapper converts the users query into one that the source can execute. To do this it adopts a simple self - describing object model. The most complicated component of the proposed solution is the mediator that represents a data access point. A mediator contains query processor and the information to process specific types of information, thus it might refine the answers returned form the data sources.

The federal database management system (FDBMS) [66] has been created to address the problem of integration of existing heterogeneous and autonomous databases. This system supports all the necessary operations for performing queries over the heterogeneous integrated sources. Heterogeneity has been defined as: difference in structure, difference in constraints and difference in query language. In this system, a data processor is a complicated component, that contains four different sub components each performing different functions: transforming processors (performs language translation), filtering processors (performs syntactic and semantic constraint application to ensure consistency), constructing processors (merges data), and accessing processors (command execution). FDBMS basically perform the following operations: create data model, perform queries and global transactions, and global access control. Similar to the CIM [38] provides three layered architecture to specify semantics.

In [61] the authors proposed a four layered data source integration architecture that contains: local schemata, local object schemata, global schema, and global view schemata. A methodology has been proposed for the architecture to integrate data from

heterogeneous data sources. The mapping is being done by acquisition of semantics of a local schema which is mostly meta - properties and their values. The authors showed how to check the consistency of the derived global schema. The proposed methodology defines mapping between global and local schemas. Thus, the the unified global database can be populated.

In [41] we are given an overview of the existing solutions to address the problem of semantics for heterogeneous sources. The author provided an overview of existing architecture for data source integration. The author shows that a lot of work that has been done assumes that the mapping has been defined by a user, although they cite some work that uses schemas and ontologies to define mappings.

2.3 Multidimensional Data Bases constraints

Multidimensional databases are central repositories of integrated data from one or more disparate sources. They store current and historical data and are used by business users for creating reports and making decisions for business users. Examples of reports could range from annual and quarterly comparisons and trends to detailed daily sales analyses [70]. The original idea for designing denormalized fact tables and dimension tables for describing data from different perspectives was pioneered by General Mills and Dartmouth College in the 1960s. This idea was later commercialized by A.C. Nielsen in the 1970s. The main idea of this design was to represent the data in an understandable way so users could see it as a fact described in the most expressive way [48].

The conceptual integration model takes this idea further and helps users to formulate their needs at a higher conceptual level.

In this thesis we are developing a mapping visual language that would help to map conceptual schema to storage sources. The mapping is defined as a query over the sources to map it to a conceptual schema that can be represented as a set of Datalog sentences. To fully capture the semantics for Conceptual Schema there are several types

of constraints to consider. We will describe what has been done to address the problem of multidimensional nature of the data warehouses.

A data cube is a data structure that allows to process information fast. It can also be defined as the capability of presenting data from different angles. Data cube consists of numeric facts called measures that are contain information from different perspectives using dimensions. Facts can be presented at different levels of hierarchical categories. Consequently, in what is following, we will consider two types of constraints to maintain relational integrity and multidimensionality for fact relations at different levels of hierarchy. We will begin with the latter ones and describe the remainder in the subsequent section.

Most multidimensional models have limited support in terms of constraints to ensure that queries are decidable at any level of hierarchy. There are several papers [42–44,48,50, 52] on the topic of heterogeneous dimensions to address the problem of summarizability.

Summarizability is a very important factor in OLAP query processing as it plays a crucial role in precomputing cube views and using them for effective query answering or obtaining another view at a higher granularity level.

In [48], there have been efforts to define the query decidability by providing semantic conditions that are required to ensure summarizability (e.g. [43]) but a lack of specifying other constraints to facilitate users logic (like EGDs and TGDs). Moreover, in this paper the authors performed rolling - up and drilling - down operations. Several techniques for checking that dimensions are summarizable have been investigated in [42]. We will briefly give an overview of some of these works and then switch to the problem of defining conceptual dependencies to express users requests.

In [43] the authors define a set of integrity constraints that allow to ensure summarizability in heterogeneous dimensions. The authors define heterogeneous dimensions as the situation where several dimensions represent the same conceptual entity, but with different categories and attributes. Moreover, the paper defines frozen dimensions, which are the smallest homogeneous dimension instances, logically separated but combined in

the same heterogeneous dimension. Frozen dimensions are defined for testing the dimension summarizability. The authors provided an algorithm that defines a subset of a hierarchical dimension and tests whether each of the categories in a hierarchical level induces at least one frozen dimension in the dimension schema.

In [50], the authors first show that a data set may be summarizable considering only some of the dimensions. An attempt to define a certain condition has been made in order to ensure the summarizability, as well as an example showing that this condition is sufficient. In particular, they showed that members of each category in each hierarchy must be disjoint, i.e. partitioned. As the authors state, this condition can tested by examining the semantic information, which essentially means consulting other logical conditions described in [53]. It is also necessary to test whether the grouping of category members is "complete". The term "complete" in this context means that the members form a logical group.

In [52], it is suggested to separate the design process into stages. The authors argue that understandability is among the most important properties in conceptual modelling and propose the initial stage of the design that ignores summarizability problems. In their algorithm the data should be normalized only at the second stage. The purpose of this stage is to transform the designed model into the one that is summarizable due to applied constraints. In the first stage, the authors defined fact - dimension relationships in a conceptual model and ignored the summarizability problem. Then they defined normalized conceptual models, which are constrained to those fact - dimension relationships that do not violate summarizability.

In [8] the notion of minimal summarizability sets were defined which is similar to [43]. The paper states that summarizability sets of categories is exactly what we need in order to choose the right set of precomputed cube views to answer the original query. Once summarizability sets have been computed, they can be used for different queries with different levels of granularity. The authors provide an algorithm to compute this summarizability set.

In [4], the authors show the constraints to ensure that we will be able to roll up and drill down for prestored views. They enriched the semantics with extra constraints to ensure that the data makes sense. The relevance of introducing and having intra- and interdimensional semantic constraints was also shown in this paper. This was the first example of the authors attempting to combine different types of constraints to create more extensive multi - dimensional model semantics. In this thesis, we make another attempt to complement EGDs + TGDs with additional constraints that helps to capture multidimensionality with internal semantics of fact/category and category/category relationships.

In [9] the authors distinguish several types of summarizability, namely, local and global, where the latter case is relative to all categories. This category of summarizability is called strict summarizability. The authors argue that the strict - summarizability conditions are too restrictive. Imposing them might lead to filtering out a wide range of useful models and instances. As an alternative to address this issue a new model has been proposed that allows us to produce extended multidimensional models.

2.4 Data source integration and query processing

It is well known that considering TGDs and EGDs together leads to undecidable issues related to query answering. It is therefore useful to identify classes of constraints which do not have harmful interaction between those constraints. Much research has been done to address this issue.

In [23] the authors tackle the problem of query answering in the presence of linear TGDs, that is, TGDs with a single body - atom, and general EGDs. Linear TGDs and EGDs as a special case, are able to capture DL - lite family logic. The syntactic condition of non - triggerability between a set of EGDs and a set of general TGDs was defined in this paper which helps to assure its separability and reduces the computational complexity. Such a condition ensures that EGDs are never triggered during the chase

construction, thus, ensuring separability. The combined complexity of query answering in the case of two sets of EGDs and linear TGDs dependencies is $PSPACE$ - complete.

The paper [64] was the first to consider the NP complexity for the query containment problem with respect to TGDs together with EGDs. Johnson and Klug proved that, under certain subclasses of EGBs and TGBs, if there is a containment between two conjunctive queries ($q_1 \subseteq q_2$) it can be verified by testing the homomorphism between these two queries. The authors propose an algorithm that maps the body of q_2 to the chase of the body of q_1, and the head of q_2 to the head of q_1. It has been shown that, in order to test the separability of CDs it is enough to test the containment of conjunctive queries under TGDs alone or with EGDs dependencies (an example of these two dependencies would be FK and PK). Moreover, it is sufficient to consider a finite, initial portion of the chase.

However, there is no specific set of IDs defined which can be non - key - conflicting, therefore the decidability of query answering cannot be deduced from this work. Although, in other papers (e.g. [25]) authors define an algorithm for query rewriting and use conjunctive query equality testing to prove that a new query (with a special subset of CDs) is equal to the base query.

In [27], the authors address the problem of checking the containment of one query in another with respect to the constraints specified in a schema. The authors showed that checking containment is generally a decidable problem and its complexity is exponential in the number of variables and constants of q_1 and q_2, and doubly exponential in the number of existentially quantified variables. The paper gives a result for query decidability on containment of conjunctive queries with regular expressions.

Papers [19, 20] consider an extended entity - relationship formalism that includes relationships and constraints such as is-a among entities and relationships, mandatory and functional participation of entities to relationships as well as mandatory and functional attributes. The authors deal with IDs together with KDs and show its separability in order to address the problem of query decidability. FO - rewritability of this semantic

is defined by checking graph - based conditions. The problem of establishing whether $t \subseteq ans(q, DB, D)$ under the sound semantic was proven to be AC_0 complete with the respect to the combined complexity and PSPACE in data complexity. The PSPACE hardness is shown by applying the given separability characteristic and assuming the fact that query answering under IDs only is in PSPACE. The PSPACE hardness is proved by reducing the problem from the finite function generation problem. The AC_0 data complexity is proven by showing that EER schemata is FO - rewritable. The authors extended the semantic by adding negative constraints to CDs that have been proven not to increase the complexity of query answering.

In [18] the authors have made the first attempt to deal with source integration under conceptual dependencies in GAV. This paper presents a technique for query processing in the case of global schema being presented as a view to map it to sources. IDs and KDs are expressed on the global schema and conjunctive queries posed over the global schema. In that paper the authors show that the tuple is the answer to the query if and only if it is the answer in the canonical database that can represent all the databases that satisfy the global schema. Therefore, the authors provided an algorithm to find the answers to a query over the given canonical database without building it.

In [25] the authors considered the GAV approach for source mapping, which was claimed to be the most used in the context of data integration. A relational data integration model has been defined, where the conceptual dependencies were formulated for the global schema. The authors have provided an algorithm for query rewriting to impose conceptual dependencies in order to avoid chase construction. A special class of dependencies was defined called non - key - conflicting IDs, or simply NKCIDs to assure IDs and KDs separability. This special class of dependencies includes simple FKs and PKs constraints. Decidability and complexity results were also presented in the paper.

In the milestone study [69], the data and expression complexity have been studied for different relational query languages. Relational calculus and its extensions by transitive closure were studied by bounded and unbounded looping. The authors discovered the

pattern of expression complexity of the investigated languages. This important paper helped us to reduce the problem of query answering over multidimensional schema to the problem of Turing Machine in a way that we can present each tuple as a set of encoding rules and position coordinates.

In [51], the data integration system has been studied from a different perspective based on the mappings between source schemas and global schemas. In the global - as - view approach, the sources contain the data and global schema can be presented as a view of the underlying sources. The second approach, called local - as - view, indicates that global schema is smaller and sources are represented as a view over the global schema. The goal of the paper is to discuss these two different approaches and provide relevant explanations to perform sound conclusions.

In [26], the authors address the problem of answering conjunctive queries when the data is incomplete considering the sound semantics. An algorithm, similar to the one in [25] was defined. Again, the constraints were imposed in a query which helps to reduce the number of conjuncts along with avoiding (possibly) infinite chase construction. The authors claim that unlike, the technique of [27], which does not give any algorithm that can be used to check containment, their technique gives a direct tool for query answering that, under certain conditions on the data, renders low computational complexity.

The paper studies variants of Datalog ontologies that are applied for tractable ontology - based query answering. A family of Datalog ontologies was presented which helps rule expressiveness in order to achieve tractability. The goal of this paper is to consider variants of Datalog±. The authors then defined a correlation between database theory and DLs.

A significant step towards tractable query answering was made in [32] where they introduced a new DL - Lite family. This ontology is expressive enough to capture the semantics of our CIM and, moreover, query answering in DL - Lite has the advantage of being first - order rewritable. This means that the conjunctive query along with constraints can be rewritten as an SQL query over the original ABox. This was the first

paper where the authors presented the algorithm for answering unions of conjunctive queries posed against ABox expressed in DL - Lite. For the given algorithm, the complexity was proven to be polynomial with respect to the size of the whole knowledge base.

The work of [22] was one of the first to define a new class of ontologies for which query answering is decidable and the data complexity (the complexity with respect to the size of the data only) is relatively low. This new class belongs to the Datalog family of languages and yet is more expressive. In particular, this language defines a sticky sets of TGDs (a sticky sets of TGDs are set of TGDs with a restriction on multiple occurrences of variables in the bodies). In this paper the authors prove the complexity results for answering conjunctive queries over sticky TGDs. The results was that queries can be first - order rewritten, and thus, rewritable into SQL. This paper was a significant step forward in defining a semantics for multidimensional schemas; an area that has been the focus of lot research. As in case of [53] the author used weakly - sticky Datalog$\pm$ because of its simple syntax and it's well defined semantics. This ontology is expressive enough to capture the kind of joins needed to capture dimension navigation. However, in this paper we are considering a more general case of linear Datalog.

In [46], the authors define a set of decidable classes of dependencies to maintain integrity. The constraints are defined to build the canonical data model (kernel model) which is a unified model. This paper makes an attempt to identify a decidable set of dependencies. More specifically, the paper considers a weakly - acyclic class of TGDs for the GLAV schema mapping fashion. This GLAV mapping specification is a hybrid of GAV and LAV, and helps to convert LAV into GAV.

In [59], the authors give a semantics for SPARQL, a graph - matching query language. Given a data source D, a query consists of a patterns which is matched against D, and the values obtained from this matching are processed to give the answer. The authors present compositional semantics for the query language and show that there is a normal form for graph patterns. The complexity of query answering of SPARQL general graph

patterns is PSPACE - complete.

In [74], the authors have defined a query processing method. The paper analyzes the query processing with respect to the GLAV method of mapping conceptual schema and sources. The ontology query language, named SPARQL, was shown to be efficient for the integration system based on mediator/wrapper. The SPARQL query driven the GLAV mapping method designed perform the transformation from the global query to the sub - queries on local ontologies.

In the [55], the authors give an overview of existing query processing algorithms over heterogeneous sources. The authors describe the heterogeneous data integration problems that can be classified as matching and mapping. Also, the authors present several strategies for solving it such as creating a 3 - layered architecture or applying a peer - to - peer architecture of data integration system. The authors made an overview of different query languages to formulate users' query. We will provide an overview of them below.

In [40], the authors presented algorithms for querying the data using materialized views that are specific for the current data integration. The materialized views are used to reformulate conjunctive queries over the global schema terms.

In [60], the authors introduces a MiniCon algorithm for answering queries using views in data integration. They claim that GAV and LAV are insufficient way of mapping specification so they created a set of Schema requirements to perform bottom - up mediated schema creation algorithm, and show how to rewrite queries over that schema.

In [31] the authors consider conjunctive queries (CQs) specified over ontologies expressed in description logic (DLs), and study the data complexity of query answering. The authors extended a well known paper [69], where the notion of complexity has been defined and it was proven for different query languages. The authors give a very useful characteristic for proving the the query in FO - rewritable so the complexity is significantly reduced to AC_0. The authors also show that for some DLs the complexity might go beyond. For this purpose, they show that for some queries the complexity becomes

NLOGSPACE - hard and PTIME - hard.

The paper [49] defines inconsistency - tolerant semantics which relies on the notion of repair of a DL ontology. The authors prove that the complexity of conjunctive query answering expressed over DL - Lite$_F$ ontologies is coNP - complete w.r.t. data complexity.

In [10,32], the authors investigate the complexity of different extensions of the original DL - Lite logic as it has become very popular to use it for description logic and semantics web. They gave a brief overview of logic of the extended DL - Lite family and presented an overview of basic characteristics. They discussed the complexity results obtained for each extension of the DL-Lite family. They also showed the complexity of knowledge base and also proved the data complexity of query answering.

In [17], the authors showed that in a classic situation when you have a global schema that contains primary and foreign key constraints, query processing is most likely facing the problem of querying an incomplete database. Also they proposed an algorithm to answer conjunctive queries posed against the global schema considering GAV mappings. The authors present an algorithm, where they rewrite a query into a set of conjuncts with respect to the set of constraints. In the case of GAV mapping and sets of TGDs and EGDs the complexity was proven to be AC_0.

In [15], the authors first define the notion of consistent query answering with respect to the global notion of inconsistency. They investigated a situation of query answering over the global schema where integrity constraints might be violated. They showed an approach illustrating how queries can be answered in consistent and inconsistent information systems by showing the application of classic and minimal logic respectively.

In [6], the authors define a notion of consistent answer in a relational database that may violate given integrity constraints. They discuss possible repaired versions of the database. They present a method of consistent query answering with respect to knowledge base for the domain.

In [36] the authors define an algorithm that creates a universal solution among all

the existing solutions. They showed that a universal solution has the exact data that is necessary for data integration and answering a query. This procedure of creating a solution is called chase. Chase is created by exhaustively applying TGDs and EGDs. This procedure can lead to its failure or it can be an infinite. The authors identified the condition that guaranties the success of the chase computation. They proved the computational complexity of computing query answers over such a universal model.

2.5 Data Warehouse semantics

In [53] the authors propose an ontological representation of a multidimensional model and mechanisms for data quality assessment via dimensional navigation. The authors extended the work of Hurtado and Mendelzon [43] and Andreas' Cali work on *Datalog*± in order to be able to perform drill - down and roll - up operations for multidimensional schemas. The extension includes categorical relations associated to categories at different levels in the dimensional hierarchies. This is the crucial point that helps to capture heterogeneous and homogeneous dimensions and assure its summarizability. The extension also considers dimensional constraints and dimensional rules that have been considered in [24] and other works. These constraints are applied for dimensions as well as fact relations.

The work of [4] also defines the semantics using DL - Lite ontology and tries to adopt it to a multidimensional schema in order to capture multidimensional navigation. For this reason the authors defined semantics for dimensions and than fact relations. Such constraints were introduced as inter- and intradimensional semantics constraints respectively. Interdimensional constraints are aimed to restrict certain values for different categories in a dimension and fact relation. In this paper the authors focused primarily on the introduction of contextual dimensions for navigating the data required. Comparing to our thesis, which focuses on defining an algorithm for query processing for multidimensional data.

In [16] the authors provide the semantics for processing the query in OLAP over imprecise data. This is the first example to address semantics issues specific to imprecise data where there is a problem of maintaining consistency between sets of queries. The semantics was given for a specific example in the form of $\Phi_1 \Rightarrow \Phi_2$, where Φ_1 and Φ_2 are conjunctions of atoms. The algorithm to decompose an imprecise database into independent components was shown using relational optimization techniques. The complexity of the algorithm has not been proven.

In [56] an algorithm for mapping layers of the conceptual integration model has been created. Similar to what we have here, the algorithm discovers mappings as complex views over the multidimensional storage model. The proposed algorithm for refactoring dimension models into summarizable models helps to avoid extra constraint definition to ensure summarizability.

Clio project [35] is a joint venture of the IBM Almaden Research Center and the University of Toronto, launched in 1999. Clio's goal was to simplify the information integration process just like the CIM. Clio pioneered the use of schema mappings which are, specifications that describe the relationship between global schema and heterogeneous source schemas. Clio defines non - procedural schema mappings to map target schema to heterogeneous source schemas. The relationship definition between elements means a business user is obliged to draw lines between schema. The process of integration is defined as the process where the target schema element should be populated with values from the source schema element. To formally express this process the inter - schema inclusion dependency, or more generally a source - to target tuple generating dependency, has been defined. The mappings are done in a sound GLAV fashion when the local - as - view approach is being represented as global - as - view.

The paper [3] describes the ADO.NET entity Framework as a part of the released .NET Framework, which is another mapping framework that is aimed to reduces mismatching of applications and data - centric services. The architecture of ADO.NET Entity Framework consists of data source - specific providers, EntityClient provider, ob-

ject services and other programming layers, metadata services, as well as design and metadata tools and services. The entity data model is based on classic relational model and borrows some concepts from the entity - relationship (ER) model. The idea is based on entities and associations. As in the ER model and our work, entities represent identified objects (in our case they are categories), while associations describe the relationship of two or more entities. EDM can be applied to multiple database management systems (relational and non - relational). The EDM together with its language - Entity SQL represent data models and can be useful for data - intensive services such as reporting, business intelligence, replication and synchronization. It can model and manipulate data at a level of structure and semantics.

PhD. thesis [45], fills some gaps in the data warehouse integration problem. The paper presents the algorithm for query processing. A query processing architecture for pipelined query execution has been presented that incrementally consume data and produces tuples that are fed into a pipelined query plan.

Hibernate [54] is a free framework distributed under the Lesser GNU Public License (LGPL). Hibernate is a framework provided in Java programming language that helps to treat database as a set of Java objects. These Java objects are Plain Old Java Objects (POJOs). The only strict requirement for a persistent class is a no argument constructor. The access to these object is provided by standard getters and setters methods. Hibernate provides data query language to facilitate any data update or manual user mapping between Java object and database. Java classes are mapped to database tables using a configuration XML file. Hibernate plays the role of connector between database and application and does mappings in order to manipulate the database. It has the advantage of being database independent and the database can easily be changed without changing the code of the application significantly.

Cognos8 [37] or IBM Cognos Business Intelligence is a web - based, integrated business intelligence product. It serves for reporting, analysis and monitoring of events and metrics. It consists of different components to meet different requirements. Query Studio

- a simple component for creating basic reports. [1]. Report Studio - used to create and manage more advanced reports. Framework Manager is a Cognos8 modeling component for building business-related metadata for BI applications. It lets modelers manually build hierarchies to allow roll up and drill down operations. To compare Clio and Cognos8, the former performs mappings specified by users manually.

In the [47] the authors have defined an integrated view over the data using the drill-across operation over datasets. They also showed how to merge cubes using merging correspondences and analyzed the complexity of generating the global cube. It is worth noting that in the worst case scenario, the algorithm used to generate all derived data cubes to answer a query over the global cube may not terminate.

In [30] the authors define dimensions and introduce dimension graphs to capture the constraints induced by the hierarchy. They do so by specifying the dimension graph and designing a query language for this framework. In contrast to our work , the user can optionally specify parent-child relationships between categories. Moreover, the authors address the problem of query decidability. In order to evaluate the query which is done in two stages, the algorithm, first, allows identification of an unsatisfiable query by checking dimension summarizability. On the other hand, the problem of EGDs and TGDs interaction is not addressed. They define a data integration method by global conceptual schema creation.

In this chapter we gave an overview of the literature regarding the topic of this thesis. Authors made a great contribution and in the following chapters we will be referring to these papers. In the following chapter we describe an architecture of the CIM and define the semantics for the global conceptual schema.

Chapter 3

Conceptual Integration Modelling Framework: Semantics

In this chapter we will briefly describe the architecture defined in [62]. We will define the semantics for the CVL in terms of Datalog sentences by presenting it as a set of relations plus certain set of dependencies (dimensional constraints).

We concentrate on mapping language (the MVL in Figure 1.2) for the mapping visual model. Inspired by [18], we clarify the analogy between two models in terms of constraints and explain why we can borrow certain constraints to map conceptual schema and multidimensional relational sources. Further, we show the differences and define additional dependencies to capture multidimensionality and ensure hierarchical summarizability.

3.1 Conceptual Integration Model Overview

The Conceptual Integration Model framework (CIM) [62] aims to integrate data from different heterogeneous sources in a way that the conceptual schema and data are driven only by information to be integrated and not by user - oriented architecture or mapping specifications.

In other words, a user will be able to specify their needs in a user - oriented language,

pose these queries against the global conceptual model, and the system will map this model to a preexisting logical multidimensional representation. At run time, a system will be able to create views over the logical multidimensional representation according to specified mappings. The main idea of the CIM is to focus on how the user can conceptually abstract from the existing information stored in sources, and how this conceptual model can be mapped to the logical multidimensional representation [62].

CIM Visual Model (CVM)	CIM Data Model (CDM)
CVL : Conceptual Visual Language	CDL : Conceptual Data Language
MVL : Mapping Visual Language	MDL : Mapping Data Language
SVL : Store Visual Language	SDL : Store Data Language

Figure 3.1: The architecture of models of the CIM Framework

The CIM framework is an approach to help users to raise the level of abstraction by allowing them to specify their needs at a high conceptual level through user - oriented conceptual visual language. The CIM framework consists of two parts that are illustrated in Figure 3.1 [62]. The first part is called the CIM visual model (CVM) which is comprised of three distinct layers: (1) the EER model that is widely used for the data warehouse design such as MultiDim [34] and StarER [68]; (2) a UML - like visual representation of the relational multidimensional model that represents data sources; and (3) a visual mapping language defined in terms of lines between corresponding elements. It is used for translating the conceptual model into the multidimensional model. The latter has been defined as a set of views over the sources (GAV approach). To construct queries we define a set of constraints to impose upon it and thereby guarantee its satisfiability.

3.2 CVL Syntax

An EER formalism is flexible and expressive enough to describe the build units of our CVL. The EER schema is built on top of the basic ER to capture certain types of relationships such as is-a relationship between entities and relationships, and mandatory and functional participation of entities in relationships.

In this section we describe an example of the CVL representation using a data warehouse describing sales (see the Figure 3.2). We shall give a syntax for each element according to what has been done in [62]. We slightly extended the classical EER defining fact and dimension category.

Example 3.2.1. *We have a multidimensional conceptual schema with the fact relationship represented as shadowed diamonds (i.e., sales); three dimensions of the fact relation are represented by shadowed rectangles (i.e., Products, Data, Workers); and levels of hierarchy are represented by unshadowed rectangles (e.g., Time, Day, Month, Year). Levels and relationships may have attributes represented by ovals (e.g., Time_id, Time, Since). Relationships are represented by unshadowed diamonds (i.e., Works, Leads).*

Relationships and entities (dimension categories and facts) have attributes that are represented in ovals.

For example, for the fact relationship we have the following numeric values as measures: discount - a percentage of a total price that was reduced on sale; Penalty - a percentage of a total price that was increased due to the late payment. name - an attribute for a dimension category Team_manager.

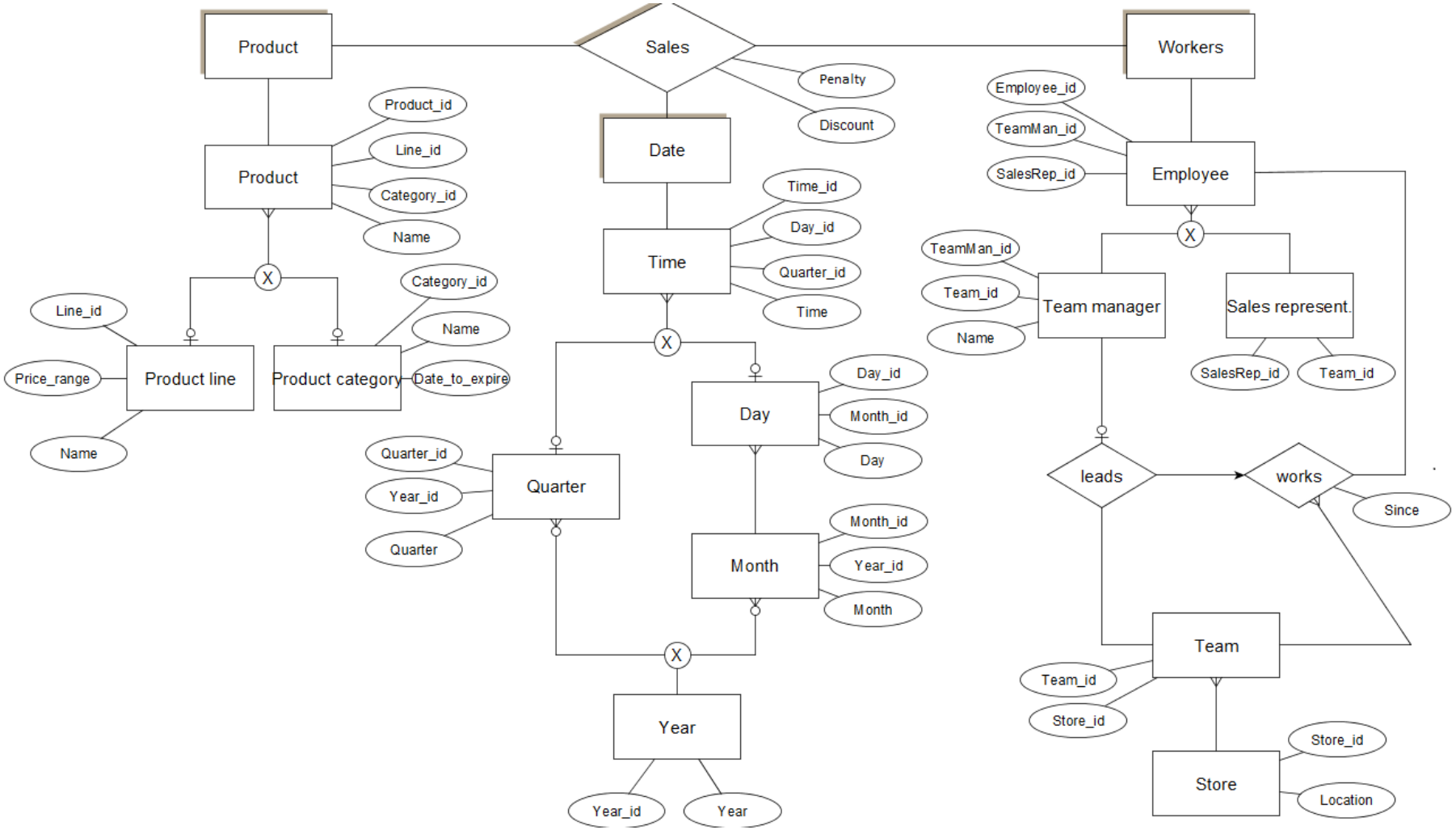


Figure 3.2: The CIM Model for Sales data warehouse

The dimensions in this multidimensional datawarehouse schema are the following:

Products. Products are categorized by line (mineral water, milk drink, fizzy drink, energy drink etc.) or by brand (Coca - Cola, Tropicana, Danone etc.).

Time. Time is structured as a particular day of the month in a year or divided in a quarters as a three - month period on a financial calendar.

Workers. This dimension is the most interesting as it includes is-a relations between entities and relations. The schema describes employees working in teams in a store, sales representatives, and team managers that are also employees, and manage teams. Managers who supervise a team also work in the same team, as imposed by the is-a relationship between two relationships.

A dimension may have an is-a relation between both entities and relationships (e.g., Sales_Represent − Employee, Leads → Works) which is represented by arrows.

Categories are related hierarchically. For example, Team is a finer category for Store. A hierarchy may have different options for a level of granularity. For example, a Day may roll - up to either a Month or a Quarter. This so - called splitting/joining parent/child relationships are expressed with parent/child relations labeled by 'X'.

3.3 MVL

As apparent in Figure 3.3, the SVL is represented as a small part of a UML - like relational data warehouse diagram. It contains a relational table, EGDs and TGDs enriched with certain constraints to maintain summarizability (conceptual dependencies, CDs). From Figure 3.3, we can see that the source contains three dimension categories - employee, team and store. They have relationships specified by foreign and key constraints. Categories are elements of dimensions, we will give a definition below.

Example 3.3.1. *Figure 3.3 shows the mapping layer between the CVL and the SVL, which specifies the correspondence between elements. It represents a CVM of the Employee dimension of the Sale data warehouse (Example 3.2.1). The rights hand side shows*

the CVL of the employee dimension and the left hand side represents a part of the source data warehouse.

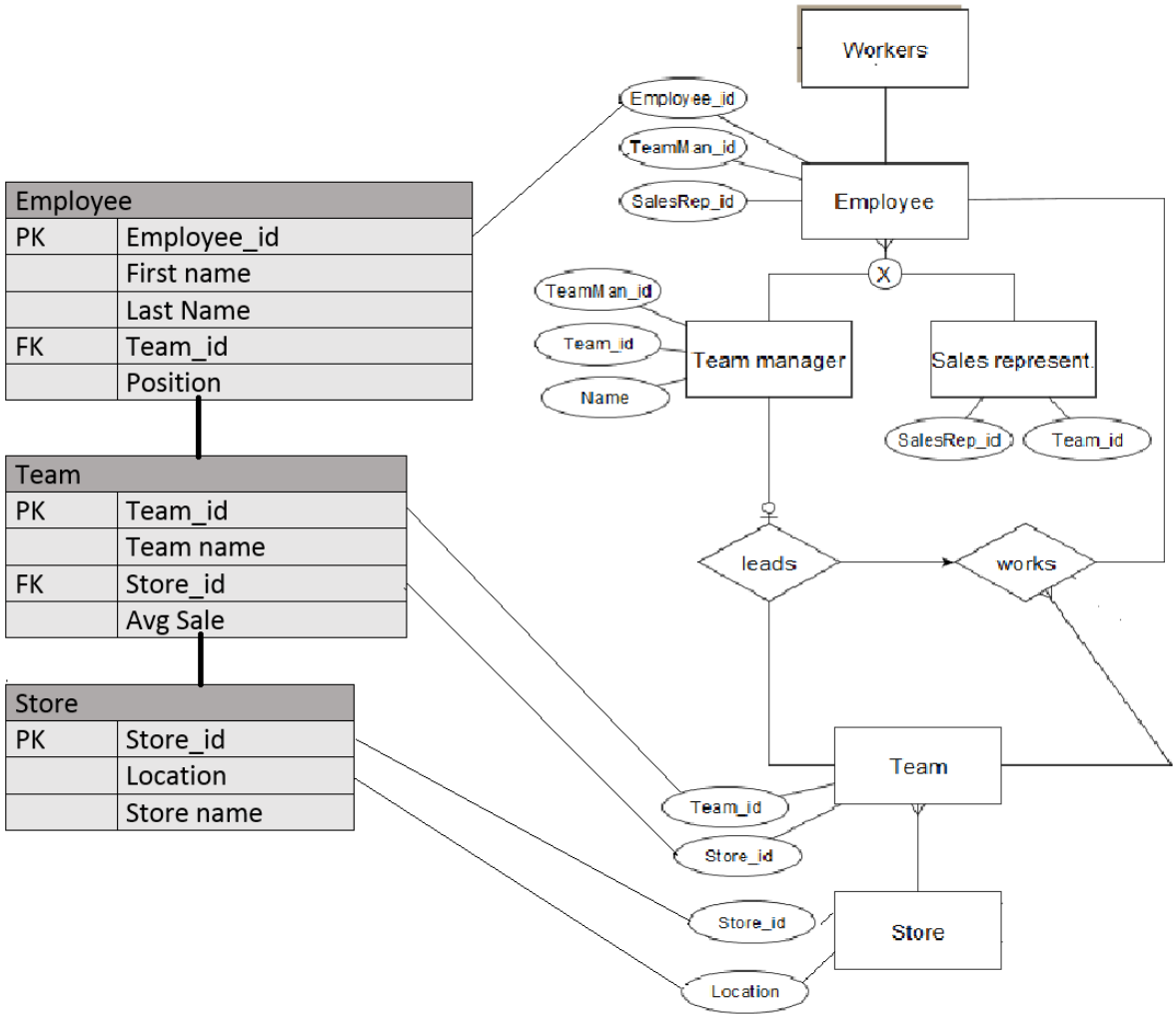


Figure 3.3: MVL model of employee dimension of the Sales Example

There is a correspondence between the CVL and the SVL models in the form of lines that define the MVL mapping. This layer helps users to avoid low - level multidimensional involvement and specifies their needs in a high - level language (CVL).

The CVL model for the employee (Figure 3.3) dimension on the right hand side has hierarchical structure (Employee → Team_manager→ Store). On the other hand in a source relational schema an employee table is denormalized and contains information

about sales representatives and team managers.

This situation is very common in a real world and forces us to specify mappings.

3.4 CVL Semantics

3.4.1 Preliminaries

We now formalize the extension of the EER model similar to [67], to adopt it for our CVL. The semantics of the CVL schema is formulated by defining a relational schema for our model along with the conceptual dependencies imposed by the CVL schema.

We introduce our semantics for the CIM by associating it to a set of relations plus a set of constraints. We will define our CIM model using Datalog as it is expressive enough to capture our semantics.

The CIM model is based on the following principles:

- Semantical: entity, relationship and attribute types are basic units of the semantics. If other types are used, then their semantics should be defined.
- Identity: Each instance should be identified. Indeed, every instance of each component should be identified.
- Name uniqueness: Entity and relationship names should be unique for the CVM schema. The attribute names should be unique for each entity and relationship type.
- Completeness: If any type is based on composite component types, then its namespaces are the set of possible values that are used for its corresponding components.
- The criteria for the constraint definition: If a set of constraints that is added to an existing constraints set, is the set of integrity constraints (IC) then its union is an Integrity Constraint $\Sigma = \Sigma' \cup IC$. The database that satisfies this extended set of

integrity constraints will be a subset of database models that satisfy the initial set of integrity constraints.

In the following part we first give a few basic preliminaries similar to [21] and then give a definition for the elements that build the multidimensional hierarchical conceptual schema.

We define the following pairwise disjoint (infinite) sets of symbols: a set Γ of constants which constitute the knowledge base, and a set Γ_f of fresh nulls, used as placeholders for unknown values.

A relational schema R (or simply schema) that represents relational sources is a set of relational symbols (or predicates), each with its associated arity. It also contains fact predicate and a dimensional hierarchy.

We write r/n to denote that the predicate r has arity n. A relational instance (or simply instance) I for a schema R is a (possibly infinite) set of atoms of the form r, where $r/n \in R$ and $t \in (\Gamma \cup \Gamma_f)$. We denote as r(I) to be the set $tuples \in I$. A database D for R is a finite instance for R such that $dom(D) \in G$.

Given a set Σ of dependencies over R, and a database D for R, the models of D w.r.t. Σ, denoted as mods(D,$Sigma$), is the set of all databases B such that B satisfies the dependencies in Σ, and $B \subseteq D$.

Now, we present the CVM by defining it in terms of relational schema with constraints. While Cali et al. presented the conceptual model using the EER model, in our case, in addition to the basic elements of the EER schema, we will also add certain ones to capture the multidimensional nature of the CIM. The CIM consists of a collection of entity, relationship, attribute, dimension, and category definitions over an alphabet of symbols $\Gamma \cup \Gamma_f$. The model is summarized as follows:

- Attribute (A): for the specific set of domains, there are attributes that are defined on these domains, such that for each attribute is its specific scope of possible values(domain). type is defined by its attributes that correspond to the specific

entity. Because measure is also at attribute we will be using the same syntax for the measure as it is a fact relationship attribute.

- Relationships (R): Relationship is a correspondence between entities. Cardinality n of R defines a relationship between n components of the same type.
- Fact (F): a special type of relationship between dimensions;
- Dimension (dim): is an abstraction used to represent physical objects or processes. Each dimension has its own unique name and hierarchical category structure.
- Dimension category (c): the entity of a dimension that describes a model at different levels of granularity;

A type defines the element and sets a domain of possible values and operations on these values. It's worth noting, that the data type restricts operations $OP(T)$ that can be applied to an instance such as:

- *Aggregation* functions can be applied to absolute values and ratio values.
- Min/Max function can be applied to interval values.

The definition of base type can be extended by defining the constraints for operation $Op(T)$ according to the domain of possible values $Dom(T)$. Thus, we extend the definition:

$$T = (Dom(T), \psi(OP(T))),$$

where $\psi(OP(T))$ is a restricted set of operations that are applied to the type.

This is very important, particularly for those that may contain numerical attributes. Such types must have the following characteristics to facilitate aggregation operations: the minimum grain; natural ordering/sorting; the definition for the compare function.

If we have T as a set of base types and $Dom(T)$ as a set of possible values, then the data warehouse can be represented as

$$G = (N, Dom, Pred(A)),$$

where N is a finite set of type names; Dom is a function that defines the correspondence between type name N and corresponding set of values $Dom(N)$; $Pred(A)$ is a set of predicates.

An output of the Dom function is a set of a type

$$A = A_1 : dom_1(A_1), ..., A_m : dom_m(A_m),$$

where $dom_i(A_i) \in Dom(T)$; $A_i \subseteq N$; $A_i : dom_i(A_i)$ - is an attribute.

Tuple t of arity n that consists of elements that belong to the set of domains $Dom_i(A_i)^n$, where $dom_i(A_i)$ is a subset of possible values of a specific type. Thus, the tuple is a

$$(a_1, ..., a_n)|a_i \subseteq dom_i(A_i), i \subseteq 1...n.$$

An entity type E is defined as

$$E = (id, attr, \Sigma),$$

where E is an entity name; id is a sequence of attributes that are called keys; $attr$ is a sequence of attributes ($attr_i$); Σ is a set of integrity constraints.

It's worth noting that $attr$ is a sequence of all the attributes that can be uniquely identified by id.

Categories of dimensions can be defined as entities.

$$c = (id, attr, \Sigma),$$

where c is a category name; id is a sequence of attributes that are called keys; $attr$ is a sequence of attributes (a_i).

Consequently, there exists a function Ψ that maps category attributes to attributes of entities $attr_C \rightsquigarrow attr_E$ such that the category constraints are not violated.

Assume the existence of a function [43] $\phi : C \longrightarrow M$ that maps each category to a member. A member of a category is represented by at least one tuple and can be uniquely identified by a primary key. Moreover, given a dimension schema $dim = (G; \Sigma), Const_{dim} : C \rightarrow K$ is a function that assigns to each category c of dim the set of all possible values k such that Σ holds.

For example (Example 3.2.1),

$January \rightarrow Month$

$Steve \quad DalBello \rightarrow Employees$

$Milk \rightarrow Products$

Let's now extend the definition of an entity type. Dr. Cali in his paper [20] defined the functional and mandatory participation which means that

- Each instance of the entity has its unique value for the attribute A;
- Each instance of the entity must have at least one value of the attribute A.

If key attributes are defined, then attributes of an entity must have at least 1 value.

Relationship type has a form

$$R = (ent, attr, \Sigma),$$

where R is a name of a relationship; ent is a sequence of entities that are participating in a relationship $\{E_1, ...E_n\}$; $attr$ is a sequence of attributes that belong to the relationship $\{A_1, ..., A_n\}$; Σ is a set of integrity constraints.

It's worth noting that the relationship type uses entities that are participating in a relationship for tuple identification.

Fact F describes the specific relationships between dimensions. Dimension dim is a set of categories that belong to the same dimension.

$$dim \subseteq dom(c_1) \times ... \times dom(c_k)$$

For example (Example 3.2.1), the fact of *sale* describes the fact of purchase of a product (dimension), sold by a manager (dimension) on a particular day (the lowest grain of the time dimension).

A dimension schema includes a directed graph (DG) of categories from different levels of hierarchy, which defines hierarchy levels (i) of the category. A dimension hierarchy corresponds to an is-a relation between categories. A dimension instance consists of a set of members (instances) of each category.

Based on these definitions, a fact relationship (F) can be determined as follows:

$$F(dim_1, ..., dim_n, a_1, ..., a_j),$$

where F is a fact, a relationship between n dimensions; $dim_1, ..., dim_n$ are dimensions, where dim_i is represented by a category c_1 of the first level and $dim_i(e_i) \rightarrow \exists E_i \, c_1(E_i)$; $A_1, ..., A_j$ - attributes.

Dimensions of a conceptual model have hierarchical structure which means that the data warehouse model is based on a hierarchical data type.

Date time type is a typical basic dimension. Then we can separate categories: {days, weeks, months, quarters, years}.

Cube schema is defined as

$$C = (dim_1, ..., dim_n, attr_1, ..., attr_j, q_i, f),$$

where dim is a sequence of a dimension; $(attr_1, ..., attr_j, q_i)$ is a set of measures that define facts and help to formulate a set of queries q; F – is a set of aggregation functions $f_1, ..., f_m$

that are defined on the attributes $attr_1, ..., attr_n$. Here we abuse the fact denotation but in the following it will be clear where we use fact relationship.

As the result we have cube schema $C = (dim_1, ..., dim_n, (A_1, q_1), ..., (A_m, q_m), f_1, ..., f_n)$. A cell of cube is a non empty value which depends on dimensions and levels of categories.

A cube cell at a particular hierarchical level can be defined using the following formula

$$Selection_{c_1 \subset dim_1, ..., c_m \subset dim_m} \subset Dom(Cube)$$

Assuming that categories have hierarchical relationship and each cell at a particular hierarchical level defines the corresponding attribute aggregation, then cube is a union of all cells.

For cube, we have categories $c, c' \in dim$ and $c \to c'$ defines the hierarchical dependence. For example (Example 3.2.1), for the dimension Date the hierarchical dependence would be: $days \to months \to quarters \to years$

The conceptual schema contains entities, relationships and attribute definitions and includes a set of constraints defined on both entities and relationships.

A query is decidable if and only if a database is consistent with integrity constraints. A relational schema R consists of a set of predicate symbols each with an associated arity. Relational database D over a schema R is a set of relations with constants as atomic values.

Conjunctive query (q) [24] of arity n over a schema R, written as q/n, is a set of conjuncts which has a form $head \leftarrow body$. A *body* has a form of $conj_1(\overrightarrow{x}, \overrightarrow{y_1})...conj_m(\overrightarrow{x}, \overrightarrow{y_m})$, where for each $i = \{1...m\}$ $conj_i(\overrightarrow{x}, \overrightarrow{y_i})$ is a conjunction of atoms whose predicate are among the symbols in a set of relations for the conceptual schema and $\overrightarrow{x}, \overrightarrow{y_i}$ contain either variables or fresh nulls.

An evaluation of the query $ans(B, q)$ denotes the query answer over the database B. The query answer is the set of n - tuples. It has a form of $< t_1...t_n >$, such that, when substituting each atom c_i for x_i in the conjunctive query it returns true in every B. [6]

Given a set of CDs, we say that a tuple t is a consistent answer to a query $q(\overrightarrow{t})$ in a database instance I, denoted as $I \models q(\overrightarrow{t})$, if for every repair I' of I, $I' \models q(\overrightarrow{t})$. A repair I' of I is another instance over schema R that satisfies CDs. The main property that ensures tractable query answering over an schema is the separability notion, which occurs when TGDs and EGDs do not interact. [24]

$$Q(Time\_id,\ Product\_id) \leftarrow Sale(Time\_id,\ Product\_id),$$
$$Product(Product\_id),\ Time(14/05)$$

Exclusion constraints set the rule that two different instances (entity, relationship) on the schema will not share any values. The exclusion constraints between two predicates r_1 and r_2 is formulated as $r_1[X] \cap r_2[Y] = \varnothing$ This condition is satisfied only if for each tuple t_1 in r_1 there is no equal tuple t_2 in r_2 so $t_1[X] \neq t_2[Y]$.

Exclusion constraints exclude common values for the basic data types. This condition is satisfied for all dimension categories.

Inclusion constraints set the rule that two expressions on the schema are in a subtype association. An inclusion constraint between relational predicates r_1 and r_2 is denoted as $r_1[X] \subseteq r[Y]$. They are satisfied by a database if and only, if for two tuples t_1 in r_1 and t_2 in r_2 there is a $t_1[X] = t_2[Y]$. The inclusion dependency forms a very important class of constraints, for example foreign key constraint is a inclusion constraint. For the CVM model we assume that for each relationship type $R(comp, attr, \Sigma)$ and E_1 is a member of a sequence *comp* and ID is an identification of E_1, we have $R[E_1[ID]] \subseteq E_1[ID]$.

For example (Example 3.2.1),

$Team(Team\_id) \rightarrow \exists Team\_id\, Works(Employee\_id,\ Team\_id)$

$Employee(Employee\_id) \rightarrow \exists Employee\_id\, Leads(Employee\_id,\ Team\_id)$.

Functional dependency allows to generate equalities among tuples. A functional dependency over a relational schema R is an expression $key(X) = \{n\}$ where $X \subseteq R$. Then the functional dependency is said to be held over a relation instance r, if for all pairs of tuples $t, t' \in r$, $t[X] = t'[X]$ we also have $t[Y] = t'[Y]$. A key dependency (KDs) (for example, a foreign key constraint) is a functional dependency. KDs is satisfied on a

relational predicate r if and only if for each $t \neq t'$ and k there is a non empty subset of attributes K, where each of them appears in a sequence only once, we have $t[K] \neq t'[K]$.

For example (Example 3.2.1), for the predicate Product we have $key(Product\_name) = \{1\}$, meaning that each $Product\_name$ can be uniquely identified by the primary key $(k) = \{Product\_id\}$. For predicate Employee we have $key(Emp\_name) = \{1\}$, meaning that each Emp_name can be uniquely identified by the primary key $(k) = Employee\_id$.

A global schema G for the CIM model can be represented as a set of relations and a set of constraints Σ. $\Sigma =< \Sigma_{ID}, \Sigma_{KD}, \Sigma_{dim} >$, where Σ_{ID}, Σ_{KD} are inclusion and functional dependencies [18] and Σ_{dim} is a set of special constraints to capture dimensions hierarchy.

Chase procedure The importance of the chase procedure can not be underestimated. It was first defined in [36] for the data exchange by Fagin et al. It is a tool to construct a unified solution over which the query can be evaluated. The chase procedure includes an exhaustive application of the chase steps until it can no longer be applied. These steps add a new tuple if the TGD is applicable or changes the instance when the EGD is applicable. The procedure fails when the instance cannot be changed to satisfy an EGDs.

TGDs chase step [36] Let tgd be a TGD of the form $\sigma = \phi(x) \rightarrow \exists y\, \psi(x, y)$. A tgd is applicable to a database D if there exists a homomorphism h such that $h(\psi(x, y)) \subseteq D$ then: (i) define $h' \supseteq h$ such that $h'(Z_i) = z_i$ for each $Z_i \in Z$, where $z_i \in \Gamma$ is a fresh labeled null not introduced previously and add to D the set of atoms in $h'(\Psi(X, Z))$, if it is not already in D [20].

EGDs chase step [20] Consider a database D for a schema R, and a KD k of the form $key(r) = A$ over R. k is applicable to D if there are two (distinct) tuples $t_1, t_2 \subseteq r(D)$ such that $t_1[A] = t_2[A]$, then: 1) if there exists $i \notin A$ such that both $t_1[A]$ and $t_2[A]$ are constants of Γ, then there is a hard violation of k and the chase fails, otherwise 2) for each $i \notin A$, either replace each occurrence of $t_1[A]$ with $t_2[A]$, or vice - verse otherwise.

Example 3.4.1. *Assume that we are given the following database.*

Time_id	Time

Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread
14/05	cheese

Product_id	Product

Figure 3.4: Step 0: Initial data

The procedure of chase algorithm application is:

Step 1

$Sales(Time\_id) \subseteq Time(Time\_id)$

Time_id	Time
14/05	
15/05	

Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread
14/05	cheese

Product_id	Product

Figure 3.5: Chase: Step 1

Step 2

$Sales(Product\_id) \subseteq Product(Product\_id)$

Time_id	Time
14/05	NULL
15/05	NULL

Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread
14/05	cheese

Product_id	Product
Cheese	NULL
Milk	NULL
Bread	NULL

Figure 3.6: Chase: Step 2

Step 3

$Key(Sale) = \{Time\_id,\ Product\_id\}$

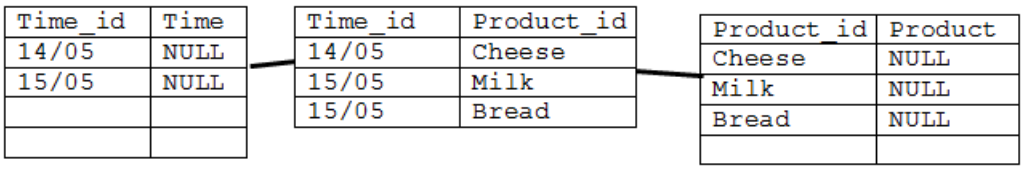


Figure 3.7: Chase: Step 3

3.4.2 Internal constraints

After we define the CVM schema by defining the basic concepts for dimensions, facts, dimensional categories, attributes and relationships, we define constraints. For each constraint we give an example from our CVL representation of the Sale data warehouse (based on Example 3.2.1) (internal constraints). We then add some more restrictions in the Section 3.4.3 [43] to guarantee dimension summarizability which means that the queries that are posed against this schema are decidable.

Our CVL model consists of a collection of entities (that are two types fact (F) and dimension categories (dim) and relationships (R) and attributes (a).

Facts

- Each relationship F has an associated predicate f of arity 1, $f/1 \rightarrow f(X)$.
  Informally, a fact of the form $f(X)$ asserts that X is an instance of entity f.
  For example (Example 3.2.1), $Sale(Employee\_id,\ Product\_id,\ Date\_id)$ asserts that $Employee\_id$, $Product\_id$, $Date\_id$ is a primary key that uniquely identifies the tuple.

- Each attribute a for an relationship F has an associated predicate a of arity 2. $a/2 \rightarrow a(X, Y)$
  Informally, a fact of the form $a(X, Y)$ asserts that Y is the value of attribute A associated with X, where X is an instance of either a fact F or a dimension dim.
  For example (Example 3.2.1), for a fact sale, $Sale(Employee\_id,\ Product\_id,$

Date_id, *Discount*) asserts a numerical attribute for the fact with the particular *Employee_id*, *Product_id*, *Date_id* defining the discount on the product.

- Each fact F involving the dimension categories $dim_1, ..., dim_n$ in C has an associated predicate f of arity n $f/n \rightarrow f(X_1, ..., X_n)$
  Informally, a fact of the form $f(dim_1, ..., dim_n)$ asserts that $(X_1, ..., X_n)$ is an instance of the fact F, where $X_1, ..., X_n$ are instances of $dim_1, ..., dim_n$ respectively.
  For example (Example 3.2.1), *Sales*(*Employee_id*, *Product_id*, *Date_id*) asserts that a fact tuple with the particular *Employee_id*, *Product_id*, *Date_id* is asserted to three dimensions - product, employee and date.

Dimension category

- Each entity *dim* has an associated category predicate c of arity 1. $c/1 \rightarrow dim(X)$
  Informally, a fact of the form $dim(X)$ asserts that X is an instance of entity *dim*.
  For example (Example 3.2.1), *Employee*(*Employee_id*) asserts that *Employee_id* is an attribute of at entity employee.

- Each attribute a for an category c has an associated predicate a of arity 2. $a/2 \rightarrow a(X, Y)$
  Informally, a fact of the form $a(X, Y)$ asserts that Y is the value of attribute a associated with X, where X is an instance of either a fact F or a another category c_i.
  For example (Example 3.2.1), for a dimension employee, *Employee*(*Employee_id*, *Salary*) asserts a Salary of an employee with the particular *Employee_id*.

- Each relationship R involving the dimension categories $c_1, ..., c_n$ has an associated predicate r of arity n. $r/n \rightarrow r(X_1, ..., X_n)$
  Informally, a relationship of the form $r(c_1, ..., c_n)$ asserts that $(X_1, ..., X_n)$ is an instance of relationship R, where $X_1, ..., X_n$ are instances of $c_1, ..., c_n$ respectively.

For example (Example 3.2.1), $Works(Employee\_id, Team\_id)$ asserts that an entities Employee and Team participate in a relationship Works. Employee works in a team.

- Each attribute a for a relationship R among the dimension categories $c_1, ..., c_n$ in C has an associated predicate a of arity n + 1.
  Informally, a fact of the form $a(X_1, ..., X_n, Y)$ asserts that Y is a value of attribute a associated with the instance $(X_1, ..., X_n)$ of relationship R. For example (Example 3.2.1), $Works$ $(Employee\_id, Team\_id, Since)$ asserts that an entities Employee and Team participate in a relationship Works and attribute Since is associated with this relationship. Employee works in a team since the date.

The intended semantics of the CVM schema is obtained by translating it into the relational model that has some constraints. Once we have defined the relational schema R for our CVM schema we add a special set of constraints: TGDs, EGDs and an additional set of dimensional constraints.

We now characterize the form of relational dependencies [21], resulting from the translation of the CVM schemata into relational schemata. Given a set of dependencies $\Sigma = \Sigma_{ID} \cup \Sigma_{KD} \cup \Sigma_{dim}$ over a relational schema R, where Σ_{ID} are TGDs, Σ_{KD} are EGDs (a special set of EGDs) and Σ_{dim} (see the Chapter 3.4.3), we say that Σ is in conceptual dependency, if we can partition R in four sets R_e, R_r , R_{ae} and R_{ar}, where R_e are predicates associated to entities, R_r are predicates associated to relationships, R_{ae} are predicates associated to attributes of entities and categories, and R_{ar} are predicates associated to attributes of relationships and fact relationship. We form the examples based on the conceptual schema (Example 3.2.1).

Table 3.1: *Fact constraints*

Definition	Constraint	Example
Attribute a for a fact F	$a_F(X, Y) \rightarrow f(X)$	$Return(Employee\_id, Product\_id, Date\_id, Return) \rightarrow Sale(Employee\_id, Product\_id, Date\_id)$
Functional attribute a of a fact	$key(a) = 1$ (a has arity 2)	$Key(Sale) = 1$
Functional dimension dim of a fact F	$key(f) = 1$ (a has arity 2)	$Key(Employee\_name) = 1$
Dimension dim in a fact F as i-th component	$f(X_1, X_2, ..., X_n) \rightarrow d(X_i)$	$Sale(Product\_id, Date\_id, Employee\_id) \rightarrow Employee(Employee\_id)$
Mandatory attribute a of fact F	$f(X) \rightarrow \exists Y\ a(X, Y)$	$Sale(Employee\_id, Product\_id, Date\_id) \rightarrow \exists Return\, Sale(Employee\_id, Product\_id, Date\_id,\ Return)$

Table 3.2: *Dimension constraints*

Definition	Constraint	Example
Attribute a for a dimension dim	$a_{dim}(X, Y) \rightarrow d(X)$	$Emp\_Name(Employee\_id, Emp\_Name) \rightarrow Employee(Employee\_id, Product\_id, Date\_id)$
Attribute a for a relationship R	$a_R(X_1, ..., X_n, Y) \rightarrow r(X_1, ..., X_n)$	$Since(Employee\_id,\ Team\_id,\ Date) \rightarrow Works(Employee\_id,\ Team\_id)$
Relationship R with dimension dim as i-th component	$r(X_1, X_2, ..., X_n) \rightarrow dim(X_i)$	$Leads(Employee\_id, Team\_id) \rightarrow Team(Team\_id)$

Table 3.2: *Dimension constraints*

Definition	Constraint	Example
is-a between relationships R_1 and R_2 where components 1,...,n of R_1 correspond to components $i_1, ..., i_n$ of R_2	$r_1(X_1, ..., X_n) \rightarrow r_2(X_{i1}, , ...X_{in},)$	$Leads(Employee\_id, Team\_id) \rightarrow Works(Employee\_id,\ Team\_id)$
Mandatory participation of a dimension *dim* in fact F	$d_i(X) \rightarrow \exists X\ f(X_1, ..., X_n)$	$Employee(Employee\_id) \rightarrow \exists Employee\_id\ Sale(Employee\_id, Product\_id,\ Date\_id)$
Mandatory attribute a of relationship R	$r(X_1, ..., X_n) \rightarrow \exists Y\ a(X_1, ..., X_n, Y)$	$Works(Employee\_id,\ \ Team\_id) \rightarrow \exists Date\ Since(Employee\_id,\ Team\_id)$
Mandatory attribute a of dimension *dim*	$d(X) \rightarrow \exists Y\ a(X, Y)$	$Team(Team\_id) \rightarrow \exists Team\_nameTeam(Team\_id, Team\_name)$
Functional attribute a of a dimension	$key(a) = 1$ (a has arity 2)	$Key(Employee\_name) = 1$
Functional attribute a of a relationship	$key(a) = \{1, ..., n\}$ (a has arity n+1)	$key(leads) = \{1\}$

Table 3.2: *Dimension constraints*

Definition	Constraint	Example
is-a between dimensions dim_1 and dim_2	$dim_1(X) \rightarrow dim_2(X)$	$Employee(Employee\_id) \rightarrow Sales\_Represent\,(Employee\_id)$
Mandatory participation of dim in R (i-th component)	$dim(X_i) \rightarrow \exists X_1, ..., \exists X_n r(X_1, ..., X_n)$	$Team(Team\_id) \rightarrow \exists Team\_idWorks(Employee\_id, Team\_id)$
Functional participation of dim in R (i-th component)	$key(dim) = i$	$Key(leads) = 2$

The semantics of the CIM schema C is defined by the association of relational schema R to it and specifying conceptual dependencies from Table 3.2, 3.3 above. As it will be shown below, this semantics (based on Dr. Cali's work [20]) cannot fully capture the semantics of the CVM. In the following, we show the examples for the defined set of CDs.

The dependencies in Σ have one of the following forms:

1. $d_1(X) \rightarrow d_2(X)$ where $\{d_1, d_2\} \subseteq R_e$

   For example,

   $Employee(Employee\_id) \rightarrow Sales\_Represent\,(Employee\_id)$,

   $Product(Product\_id,\ Category\_id) \rightarrow Category(Category\_id)$,

   $Day(Day\_id,\ Month\_id) \rightarrow Month(Month\_id)$

2. $d(X_i) \rightarrow \exists X_1, ..., \exists X_n\, r(X_1, ..., X_n)$, where $d \subseteq R_e$ and $r \subseteq R_r$

   For example,

$Team(Team\_id) \rightarrow \exists Team\_id\, Works(Employee\_id,\, Team\_id)$

$Employee(Employee\_id) \rightarrow \exists Employee\_id\, Leads(Employee\_id,\, Team\_id)$

3. $r(X_1, ...X_n) \rightarrow d(X_i)$, where $d \subseteq R_e$ and $r \subseteq R_r$

   For example ,

   $Works(Employee\_id,\, Team\_id) \rightarrow \exists Date\, Since(Employee\_id,\, Team\_id)$

4. $r_1(X_1, ..., X_n) \rightarrow r_2(X_{i1}, ..., X_{in},)$, where $r_1, r_2 \subseteq R_r$ and also $[i_1, ..., i_n]$ isa permutation of a set[n]

   For example,

   $Leads(Employee\_id,\, Team\_id) \rightarrow Works(Employee\_id,\, Team\_id)$

5. $a_D(X, Y) \rightarrow d(X)$, where $a \subseteq R_{ae}$ and $d \subseteq R_e$

   For example,

   $Employee\_Name(Employee\_id,\, Emp\_Name) \rightarrow Employee(Employee\_id)$

   $Month(Month\_id,\, Month) \rightarrow Month(Month\_id)$

   $Product\_price(Product\_id,\, Product\_price) \rightarrow Product(Product\_id)$

6. $d(X) \rightarrow \exists Y\, a(X, Y)$, where $a \subseteq R_{ae}$ and $d \subseteq R_e$

   For example,

   $Team(Team\_id) \rightarrow \exists Team\_name\, Team(Team\_id,\, Team\_name)$

   $Month(Month\_id) \rightarrow \exists Month\, (Month\_id,\, Month)$

   $Product(Product\_id) \rightarrow \exists Product\_price\, (Product\_id,\, Product\_price)$

7. $a_R(X_1, ..., X_n, Y) \rightarrow r(X_1, ..., X_n)$ where $a \subseteq R_{ae}$ and $r \subseteq R_r$

   For example,

   $Since(Employee\_id,\, Team\_id, date) \rightarrow Works\, (Employee\_id,\, Team\_id)$

8. $r(X_1, ..., X_n) \rightarrow \exists Y\, a(X_1, , ..., X_n, Y)$, where $a \subseteq R_{ae}$ and $r \subseteq R_r$

   For example,

   $Works(Employee\_id,\, Team\_id) \rightarrow \exists Since\, (Employee\_id,\, Team\_id,\, date)$

The dependencies in Σ_K have one of the following forms:

1. $key(r) = \{i\}$ where $r \subseteq R_r$ and i$\subseteq${1,...,arity(r)}

   For example,

   $key(leads) = 1$

   $key(leads) = 2$

   $key(works) = 1$

   $key(works) = 2$

2. $key(a) = \{1\}$ (a has arity 2) where $a \subseteq R_{ae}$

   For example,

   $key(Product\_name) = \{1\}$

   $key(Emp\_name) = \{1\}$

   $key(Month) = \{1\}$

3. $key(a) = \{1, ..., n\}$ (a has arity n+1) where $a \subseteq R_{ae}$ and n=arity(a)-1

   For example,

   $key(leads) = \{1\}$

4. $key(d) = \{1\}$ (d has an arity 2) where $d \subseteq R_e$

   For example,

   $Key(Leads) = 2$

- For each predicate $r \subseteq R_r$ and for each i$\subseteq$1,...,arity(r) there exists exactly one predicate $d \subseteq R_e$ such that $r(X_1, , ..., X_n) \rightarrow d(X_i)$ is in Σ_{ID}

- For each predicate $f \subseteq R_e$ and for each i$\subseteq$1,...,arity(r) there exists exactly one predicate $d \subseteq R_e$ such that $f(K, X_1, , ..., X_n) \rightarrow d(X_i)$ is in Σ_{ID}

- For each predicate $r \subseteq R_r$ and for each i $\subseteq$ 1,...,arity(r)
  if $d(X_i) \rightarrow \exists X_1, ..., \exists X_n\, r(X_1, ...X_n)$ where $d \subseteq R_e$ occurs in Σ_{ID} then $a(X, Y) \rightarrow d(X_i)$, there exists exactly one predicate $d \subseteq R_e$ such that $a(X, Y) \rightarrow d(X)$ is in Σ_{ID}.

- For each predicate $r \subseteq R_r$ and for each i $\subseteq$ 1,...,arity(r)
  if $d(X_i) \rightarrow \exists X_1, ..., \exists X_n(K, X_1, \ldots, X_n)$ where $d \subseteq R_e$ occurs in Σ_{ID} then $a(X, Y) \rightarrow d(X_i)$, there exists exactly one predicate $d \subseteq R_e$ such that $a(X, Y) \rightarrow d(X)$ is in Σ_{ID}.

- For each predicate $a \subseteq R_{ae}$, if $d(X) \rightarrow \exists Y\ a(X, Y)$, where $d \subseteq R_e$, is in Σ_T, then $\exists Y\ a(X, Y) \subseteq d(X)$ is also in Σ_{ID}

- For each predicate $a \subseteq R_{ar}$, there exists exactly one predicate $r \subseteq R_r$ such that $a(X_1, , ..., X_n, Y) \rightarrow r(X_1, ..., X_n)$ is in Σ_{ID}.

- For each $a \subseteq R_{ar}$, if $r(X_1, ..., X_n) \rightarrow \exists Y\ a(X_1, ..., X_n, Y)$, where $r \subseteq R_r$, is in Σ_T, then $a(X_1, ..., X_n, Y) \rightarrow r(X_1, ..., X_n)$ is also in Σ_{ID}.

Now, we will show that the EER schema enriched with the set of Σ_{ID} and Σ_{KD} constraints can not express the multidimensional nature of our CVM. We should point out that the interaction between TGDs and EGDs might be harmful. There are several papers that address this problem (e.g. [18]).

As mentioned above, facts are characterized by several dimensions. Dimensions themselves have hierarchical structure. For example, sales are characterized by the time and location dimensions that have a hierarchical structure. The base granularity for the fact is a day, employee and product, which means that the sales facts are aggregated for a particular product by the day. Precomputed cube views are stored to speed up query processing, so that the query analyzer can choose the closest prestored view and processes it to return the result.

Thus, the notion of summarizability was introduced to study aggregate navigation in OLAP dimensions [42, 43, 50, 52, 58]. A definition of summarizability was given [43] as the ability to correctly compute a single - category cube view from another pre - computed single category cube views. In other words, we should be able to perform drill - down and roll - up operations correctly. In [43], the authors outlined several

conditions to assure that dimensions are summarizable, however, they did not consider other constraints related to fact - dimension relationships. The problem is raised when facts are characterized by heterogeneous dimensions. Kimball [48] introduced the term heterogeneous products to refer to the situation, where facts have the same dimension, but they are still characterized from different perspectives.

Theorem 3.4.1. *A category c is summarizable from a set of categories C in a dimension instance dim if for every category c_i, where i is a level of the category in a dimension and $i > 1$ have dim $\vDash c_i.c_{i+2} \supset c_i.c_{i+1}$*

The intuition behind Theorem 3.4.1 [43] is that in order for the dimension to be summarizable, every member should have a predecessor in a parent category. In what follows, we show that this condition breaks without certain dimensional constraints, if we were to consider TGDs and EGDs alone.

Corollary 3.4.1. *From the Theorem we can conclude that the dimensions are not summarizable without certain conditions (given below). Thus, the query answering is undecidable considering only TGDs and EGDs.*

3.4.3 Outer Constraints

We present a set of dimensional constraints to capture the multidimensionality of the CVM. Constraints, demonstrated in [18] are not able to capture the semantics of our model and ensure dimension summarizability. Here we present a set of dimensional constraints that should be defined along with IDs and KDs for the CIM. We will give an example for each constraint based on our sales data warehouse example 3.2.1. Then we explain why ID and KD are not able to capture the CIM.

Table 3.3: *Fact constraints*

Definition	Constraint	Example
Fact constraints		
Constraint	*Example*	*Comments*
$F_i(dim_i, a_i) \rightarrow F_{i-1}(dim_{i-1}, a_{i-1})$	$Sales_{month}(dim_{date}, a_{month}) \rightarrow Sales_{day}(dim_{date}, a_{day})$	This constraint will help to aggregate facts from different hierarchies.
$F(dim, a) \rightarrow \exists e_i\, c(e_i)$	$Sales(Date, a) \rightarrow \exists Date\_idSales(Date\_id, a)$	Represents a Foreign key dependency between dimensions and fact table
$key(a) = [1...n]$	$Sales(Sales\_num) = [1...3]$	each attribute is defined by $1...n$ attributes which are FK in fact
$key(r) = \{i\}$	$key(Date) = 1$	Will let us to roll up. Functional participation category in fact relationship

Table 3.4: *Dimension constraints*

Definition	Constraint	Example
Dimension constraints		
Constraint	*Example*	*Comments*
$c_i(e_i) \rightarrow c_{i+1}(e_j)$	$Day(Day\_id) \rightarrow Month(Month\_id)$	is-a relationship between categories: non - overlapping and disjoint
$c_i(e_i) \rightarrow dim_j$	$Day(Day\_id) \rightarrow Dim_{Time}$	Each category belongs to either on of the categories
$a(e_i, a) \rightarrow c(e_i)$	$DayShortForm$ $(Day\_id,\ DayShortForm)$ $\rightarrow Day(Day\_id)$	Categories have its noncategorical attributes to aggregate
$c_i(e_i) \rightarrow \neg\exists e_i\, c_j(e_i)$	$Team(Team\_id) \rightarrow \neg\exists Sales\_Rep(SalesRep\_id)$	Disjointness
$c_i(e_i, e_j) \rightarrow \exists e_j c_{i-1}(e_j)$	$Emp(Epm\_id, Team\_id)$ $\rightarrow \exists Team\ (Team\_id)$	Non - overlapping

We will proceed to give examples for the defined constraints based on the base CIM schema in the example 3.2.1. We will prove its necessity by showing that A database D for R is a finite instance for R such that constraints that are defined in [18] can not fully describe the semantics for the CIM model as it is unable capture its multidimensionality. A table gives a counter - example that is semantically correct based on Dr. Cali's model, although it would not function in a multidimensional reality.

Table 3.5: Constraint examples

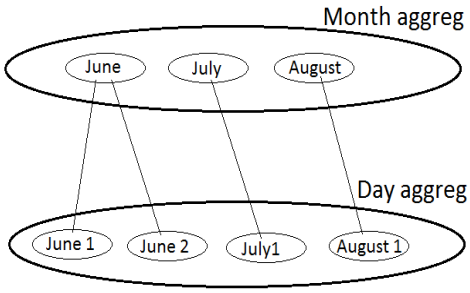
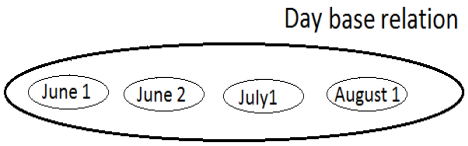
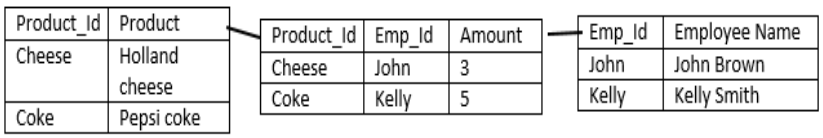
Constraint	*Example for CIM*
$F_i(dim_i, a_i) \rightarrow F_{i-1}(dim_{i-1}, a_{i-1})$ (3.5.1)	 Counter - example 
$F(dim, a) \rightarrow \exists e_i\, c(e_i)$ (3.5.2)	

Table 3.5: Constraint examples

<table>
<tr><th>Constraint</th><th>Example for CIM</th></tr>
<tr><td>$key(a) = [1...n]$
(3.5.3)</td><td>PK PK
<table><tr><th>Product_Id</th><th>Emp_Id</th><th>Amount</th></tr><tr><td>Cheese</td><td>John</td><td>3</td></tr><tr><td>Coke</td><td>Kelly</td><td>5</td></tr></table>
Counter - example
<table><tr><th>Product_Id</th><th>Emp_Id</th><th>Amount</th></tr><tr><td>Cheese</td><td>John</td><td>3</td></tr><tr><td>Coke</td><td>Kelly</td><td>5</td></tr><tr><td>Coke</td><td>Kelly</td><td>2</td></tr></table></td></tr>
<tr><td>$key(r) = \{i\}$
(3.5.4)</td><td>PK PK
<table><tr><th>Product_Id</th><th>Emp_Id</th><th>Amount</th></tr><tr><td>Cheese</td><td>John</td><td>3</td></tr><tr><td>Coke</td><td>Kelly</td><td>5</td></tr></table></td></tr>
</table>

Table 3.5: Constraint examples

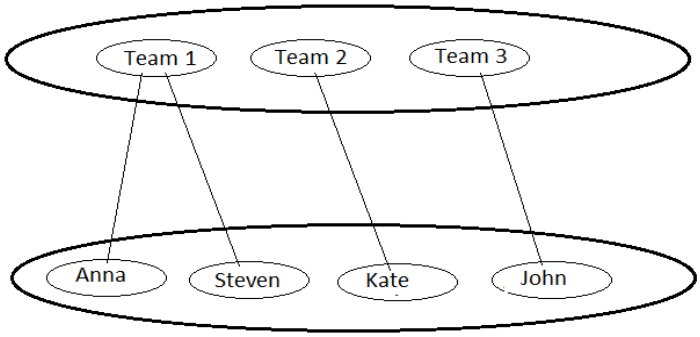
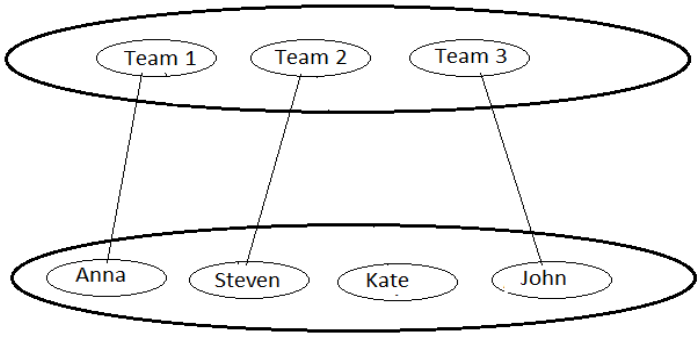
Constraint	*Example for CIM*
$c_i(e_i) \rightarrow c_{i+1}(e_j)$ (3.5.5)	 Counter - example 

Table 3.5: Constraint examples

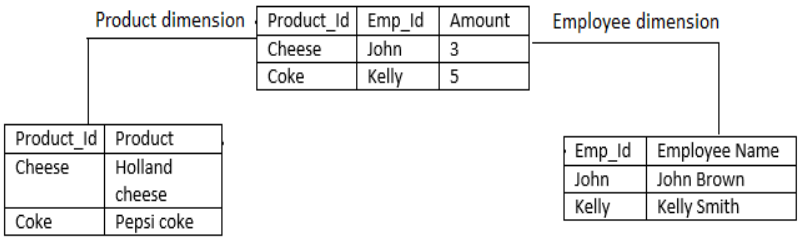
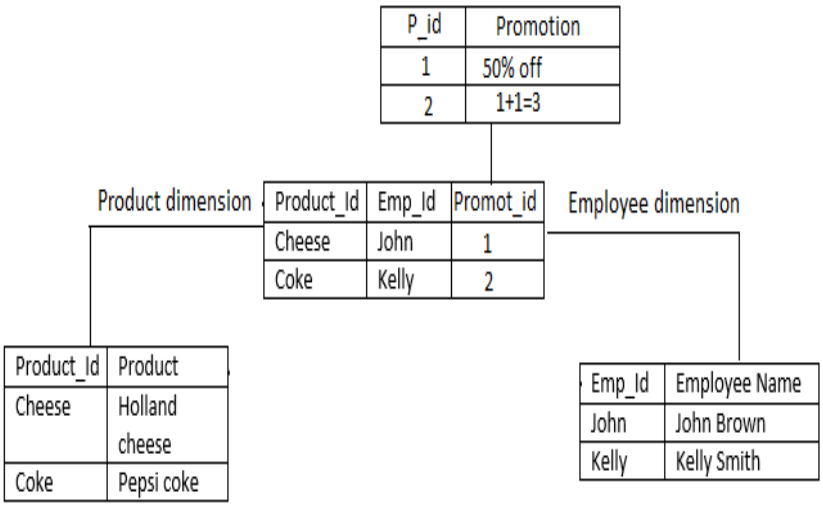
Constraint	*Example for CIM*
$c_i(e_i) \rightarrow dim_j$ (3.5.6)	 Counter - example 

Table 3.5: Constraint examples

Constraint	*Example for CIM*
$a(e_i, a) \rightarrow c(e_i)$ (3.5.7)	Employee dimension Emp_Id | Employee Name | Age John | John Brown | 26 Kelly | Kelly Smith | 48
$c_i(e_i) \rightarrow \neg\exists e_i c_j(e_i)$ (3.5.8)	Employees Emp id | Team_id | SalesR_id 1 | Anna | null 2 | John | null 3 | null | Kate 4 | null | Tom Sales represantatives SalesR_id | SalesR_Name Anna | Anna Tomson John | John Willson Team managers TeamM_id | TeamM_Name Kate | Kate Black Tom | Tom Handrick

Table 3.5: Constraint examples

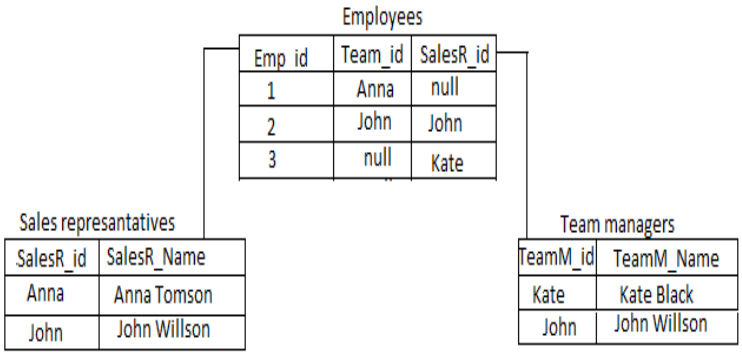
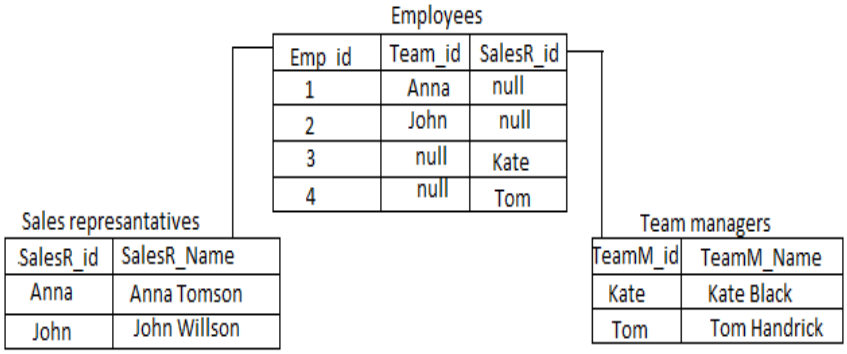
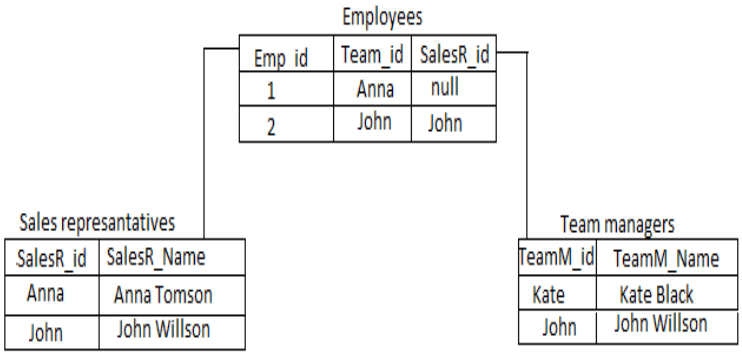
Constraint	*Example for CIM*
	Counter - example 
$c_i(e_i; e_j) \rightarrow \exists e_j c_{i-1}(e_j)$ (3.5.9)	

Table 3.5: Constraint examples

Constraint	*Example for CIM*
	Counter - example 

As it can be seen from the examples, not all relational models defined over ID and KD, would fit the multidimensional model.

The first 3.5.1 constraint helps to drill up and down along the dimension hierarchy. In other words, this constraint assures that all tuples of one hierarchical level are taken to compute the tuple of the higher hierarchy level. These precomputed tuples are stored to be returned with the user query. Month - day hierarchy can be considered as an example of this constraint. To store precomputed information regarding sales that are done in July, we have to consider all tuples containing the information about each day in July.

From constraint 3.5.2 it is clear that if there is a dimension for a fact relation, there should be at least one category for this dimension. For example, if we have product and employee dimensions for the sales fact relation, then we have one category entity for each dimension with a foreign key dependency.

Constraint 3.5.3 states that each tuple in fact relations has a composite primary key.

Constraint 3.5.5 sets the relationship between members of two categories of different consecutive hierarchy levels. Namely, if there is a member of a lower category, there should be its predecessor in a succeeding category.For example, all products should be assigned to a product category.

Constraint 3.5.6 states that all categories should belong to any of the dimensions. In contrast, constraints that are described in [18] will not prevent fact entity from having an is-a relationship with entity, that doesn't relate to any of the dimensions. For example, a promotion relation is not a dimension but it has inclusion dependency with fact relation. Thus, it is not acceptable for the CIM, but it is valid for classic EER.

Constraint 3.5.7 states that all the dimension categories should have attributes to aggregate fact tuple.

Constraint 3.5.8 sets the non - overlapping is-a relationship between parent/child categories. In other words, a member can not belong to two categories at the same time. For example, an employee can not be a sales representative and team manager at the same time.

Constraint 3.5.9 sets the disjoint is-a relationship between parent - child categories. In other words, a member must belong to at most one parent category. For example, any sales representative should be an employee.

In conclusion, in this chapter we defined the semantics for the CVL in terms of Datalog sentences. We showed that constraints that are defined in [18] cannot capture multidimensional nature of the model. We presented an examples where ID+KD dependencies are not enough. We defined a new set of constraints to capture summarizability of the dimensions.

In the next chapter we address the problem of query answering by presenting an algorithm that compiles constraints that are defined in this section into a new query so that it can be posed against sources directly.

Chapter 4

Query answering in the GAV approach

In this section we prove the complexity of the query answering algorithm in the CIM. It has been shown in many papers [18], once we know that the chase does not fail and IDs and KDs don't interact, the problem is in the highly tractable class AC_0 in data complexity. We will show that for the conceptual schema, key and inclusion dependencies are not conflicting which means the query can be answered by taking only IDs into consideration. Inspired by the work of Cali, we present a query answering algorithm that imposes conceptual dependencies in a conjunctive query so that it can be posed against relational sources directly. We will prove the decidability of the algorithm by showing the data and combined complexities.

4.1 Global-as-a-view approach of mapping

We address the problem of query answering in the global-as-view approach by presenting a perfect rewriting algorithm. The algorithm modifies the query conjuncts according to the key and inclusion dependencies that are expressed on the global integration schema.

4.2 Data integration

Given a CQ q of arity n, the evaluation of q in the database D, denoted by $eval(q, D)$, is the set of n constants such that for a tuple t $\in eval(q, D)$ a substitution of the variables in the query conjuncts with values from the D would return true. The evaluation of a union of CQ (UCQ) Q in D, denoted by $eval(Q, D)$, is the set $\cup_{q \in Q} eval(q, D)$. The set of certain answers to a union of conjunctive queries over a DL-Lite ontology, denoted by $< \mathcal{T}, \mathcal{A} >$, is the set of tuples $\cap_{I \in Mod(<T,A>)} eval(Q, D)$ [5].

Definition 4.2.1. *[63] Given a UCQ Q and a TBox T , a UCQ Q' is a* perfect rewriting *of Q with respect to T, if for every ABox A such that $< \mathcal{T}, \mathcal{A} >$ is consistent, $cert(Q, <T,A>) = eval(Q, I_A)$.*

Given a database D and a set of Σ_{TGDs} and Σ_{EGDs}, the chase algorithm for D and $\Sigma = \Sigma_{TGDs} \cup \Sigma_{EGDs}$ consists of an exhaustive application of TGDs and EGDs chase rules in a breadth-first fashion, which leads to the result of a (possibly infinite) chase for D, denoted as $chase(D, \Sigma)$. The (possibly infinite) chase for D and Σ is a universal model of D w.r.t. Σ, i.e. for each instance $B \subseteq mods(D, \Sigma)$, there exists a homomorphism from $chase(D, \Sigma)$ to B. Using this fact it can be shown that $D \cup \Sigma \models q$ iff $chase(D, \Sigma) \models q$, for every BCQ q.

First, analogously to [18] it can be proven that,

Theorem 4.2.1. *There exists a (possibly infinite) chase(D, Σ) such that, for any database instance B there exists a homomorphism Ψ that sends the tuples of the chase(D, Σ) to the tuples of B, which is a legal database w.r.t D.*

Proof. Our proof is heavily based on the evidence derived from [18]. We proceed by induction on the application of rules that are used during the construction of chase(D, Σ).

As a base step, it is obvious that chase(D, Σ) = B. For each $tuple(c_1, ..., c_n)$, if $(c_1, ..., c_n) \subseteq r^{chase(D,\Sigma)}$, then $(\Psi(c_1), ..., \Psi(c_n)) \subseteq r^B$.

Inductive step: Suppose that, without loss of generality, we can assume while applying the rule, we would need to insert the tuple $(\alpha, f_{r,i_1}(\alpha), f_{r,i_2}(\alpha))$ in $r^{chase(D,\Sigma))}$. In this case the $f_{r,i_1}(\alpha), f_{r,i_2}(\alpha))$ denote tuples of the predicate r and $r^{chase(D,\Sigma))}$ denote a predicate r in the $chase(D,\Sigma)$. Let us assume, that the tuple is inserted in the relation $r^{can(I,D)}$ because of the foreign key constraint $w[j] \subseteq r[1]$.

It is obvious, that the insertion is done in a dimension/fact relation by applying the chase for the $w[j] \subseteq r[1]$ constraint. If both, w and r are dimension relations, the evidence from [18] is straightforward.

If we assume that $w[j]$ appears in a fact relation, we must still make an insertion in a dimension relation. Because of the constraint $w[j] \subseteq r[1]$, we have that there is a tuple t in $w^{chase(D,\Sigma)}$ such that $t[j] = \alpha$, where t is in the fact relation.

Without loss of generality, we assume that the arity of a fact relation is 3 and $key(r) = \{1\}$. If this holds and since we are applying the chase rule, we have a tuple in a fact relation in a canonical model, where $t^{can}[j] = \alpha$. By the hypothesis, there is a homomorphism from this tuple to a tuple $t[i] = \Psi(t[i]) = \alpha$, and $\Psi(\alpha) = \beta$ in legal database B. Because of $key(r) = \{1\}$ and the rule $w[j] \subseteq r[1]$, this tuple is unique, thus there is a tuple (β, γ, δ) that exists in B. Further, there is a homomorphism from $(\alpha, fr, i_1(\alpha), fr, i_2(\alpha))$ to a tuple in legal database B. □

Then we show that, if $I =< G, M_{GS}, S >$ is consistent w.r.t. D, then a tuple t of constants is in $q^{I,D}$, if and only if t is in the answer to q over the database $chase(D,\Sigma)$.

Theorem 4.2.2. *Let $I =< G, M_{GS}, S >$, let q be a query posed to I, D is a source database for I, and t is a tuple of constants of the same arity as q. In this case G is a global schema that is expressed as a set of relations plus a set of IDs, M_{GS} is a mapping between sources and global schema and S sources are presented as a set of relations. If I is consistent w.r.t. D, then we can assume $t \subseteq q^{I,D}$ if and only if t is in the answer to q over $chase(D,\Sigma)$.*

Proof. For the "if" direction, this means that if $t \in ans(chase(D,\Sigma), q)$, then $t \subseteq q^{I,D}$. It

flows from the Theorem 4.2.1 where it states that there is a function Ψ such that, for each relation r, if a tuple t of a type $(c_1, ..., c_n)$ and the following holds $(c_1, ..., c_n) \subseteq r^{chase(D,\Sigma)}$, then Ψ maps each c_i to a value in r^D such that $\Psi(c_1) = \alpha$ so $(\Psi(c_1), ..., \Psi(c_n)) \subseteq r^D$.

As for the "only-if" direction, if a model B is consistent with D, then chase(D, Σ) does not violate any inclusion or functional dependency in G. Consequently, since chase(D,Σ) is a universal database for all the models B and since we obtained chase(D, Σ) from B, then $B \subseteq chase(D, \Sigma)$. If $t \notin ans(chase(D, \Sigma), q)$, then t is not in the answer of q over any legal model B_i. Consequently, $t \not\subseteq q^{I,D}$. □

In the context of conceptual data models, data may be inconsistent and incomplete with respect to the constraints that are imposed on the model. To repair the violations from the constraints, we build the chase model. The idea of the chase is to convert the initial facts into a new set of facts such that it would satisfy the dependencies. The procedure is done by collapsing facts (according to KDs) or adding new facts (according to IDs). We applied the same procedure in the Example 3.4 for the fact table as well as dimensions.

The DL-Lite family of description logic is largely used to allow complex query answering over ontologies with very large instance sets (ABoxes) and sets of dependencies (TBoxes). A distinguishing feature of the DL-Lite with respect to the other DLs is that the perfect reformulation of conjunctive queries can be expressed by first-order queries. Thus, we reduce the complexity of the query answering down to AC_0.

However, it's worth noting that the TBox should also include certain dependencies formulated for hierarchical dimensions. In other words, to formulate perfect rewriting we should take into account the dimensional summarizability.

More precisely, the query answering is performed by first building a summarizable dimensions and then rewriting the query with respect to the integrity constraints part of the ontology (TBox), thus imposing the constraints into the initial query. Such a perfect reformulation is then evaluated over the extensional part of the ontology (ABox) only.

4.3 Query answering algorithm

We present a new algorithm for the conjunctive queries reformulation over DL-Lite ontologies. This algorithm is based on the following idea: unlike previous approaches, we take into account the multidimensionality of the data warehouse using the DISMAT algorithm that was previously defined in [43]. Our algorithm DimBUILT (Algorithm 1) first builds subhierarchies based on defined IDs, and then we apply the DISMAT algorithm to tests whether each of them induces frozen dimension. A frozen dimensions enjoys an advantage of summarizability which guarantees query satisfiability.

Algorithm 1: DimBUILT(Ψ,Σ_I,q)

input: relational schema Ψ, inclusion dependencies Σ_I,
union of conjunctive queries Q
if *Q is empty* **then**
  then return()
end
else if $q_i \subseteq$ *fact* f **then**
  then for ***each*** *ID such that head contains fact* F $F(X) \rightarrow \exists c(X,Y)$ **do**
    dim_i = SEARCH(Q,c)
    if *DIMSAT(dim_i,c) == true* **then**
      ID-rewrite(Ψ,Σ_I, Q)
    end
  end
end

According to [43], a frozen dimension is a minimal set of categories that build a hierarchy. Frozen dimensions can be conveyed from the hierarchical schema. For simplicity, the authors associate a single attribute, called Name, to every category of a dimension.

Definition 4.3.1. *[43] Given a dimension schema dim, and a category c derived from it, a frozen dimension of dim with root c is a dimension instance* $f = (G, member, Name) \in I(dim)$*, such that the following holds:*

- $member_c = \{\phi(c)\}$*, where* $\{\phi(c)\}$ *a special function that maps every member to a category;*

- *each category* $c' \in C$ *has at most* $\phi(c')$ *members ;*
- *for every member* x *such that* $x \neq \phi(c)$, $\phi(c)$ *is an ancestor of* x *and*
- *for every category* c' *such that* $member_{c'} \neq \emptyset, Name(\phi(c')) \in Const(c')$.

First, for the query, the algorithm builds the dimension subhierarchies (DimBUILT Algorithm 1). This algorithm prompts another two algorithms to: 1) build a subhierarchy tree that is defined in the query dimension by its conjuncts; 2) check if the dimension is summarizable. Once we verify that the dimension structure defined for the query is summarizable, we can call the algorithm to rewrite conjunctive queries (inspired by Dr. Cali's work).

The DimBUILT algorithm detects the dimensions that are defined in the query based on the FKs constraints of the type $F(X) \rightarrow \exists c_i\,(X, Y)$, where $c_i(X, Y) \subseteq dim_i$ for the fact F relation. For each detected dimension of the fact we find the first category that is attached to it via a constraint. Then, the DimBUILT algorithm 1 calls the SEARCH algorithm 2, which builds a hierarchical tree to test the summarizability. To define the specified hierarchies, we assume that there exists a function Φ that maps each atom of the query to a category. The mapping is done in such a way that, if we have a $c_i(X)$ atom in a query, we can map it to a category of a dimension, thus $\Phi(c_i(X)) = c_j$, where c_i is an atom in a query and c_j is a category:

- if we have a query of the form $q(X, Y) \leftarrow c(X),\ c_1(X_1, X), ..., c_m(X_m, X_{m-1})$ we will be able to map it to a hierarchical path such that $c \rightarrow c_1 \rightarrow ... \rightarrow c_n$
- if we have a query of the form $q(X, Y) \leftarrow c(X),\ c_1(X_1, X), ..., c_m(X_m = k)$ and an equality atom $c_m(X_m = k)$ then k is a member of a category of c_m

The SEARCH algorithm builds a dimensional hierarchy for the query atoms based on the IDs constraint of the form $q_i(X) \rightarrow \exists X v_i(X, Y)$, where $q_i(X)$ is an atom in the query and $v_i(X, Y)$ is a node in a hierarchical tree. For each node in the tree the algorithm searches the atom to attach based on the FK constraint. The atom mentioned in the

Algorithm 2: SEARCH(Q,c)

```
input  : union of conjunctive queries Q,
         first conjunct c
output: dimension hierarchy dim
if Q is empty then
  then return()
else
    G=Graph();
    AddVertex(c);
    for each q_i ⊆ Q do
        for each vertex v_i of the graph G do
            if exists q_i(X) → ∃X v_i(X,Y) then
                Add q_i as a child to G(v_i)
            end
        end
    end
end
return G
```

head of the dependency becomes the ancestor, and the atom defined in the body that is already in the tree. Thus, all atoms that build a corresponding hierarchy are added to the tree. Once the hierarchy is built, the DISMAT algorithm is called to check its summarizability.

The DISMAT algorithm has been defined in [43] and it checks all the possible subhierarchies and tests whether it is a frozen dimension. When a subhierarchy is built, the algorithm defines constraints that are relevant to the specified schema. Then the algorithm does the renaming for every TGD, such that every atom of the constraint is substituted it with $\top$, if this atom is in a hierarchy, and with $\bot$ otherwise. So, in order to test whether a built hierarchy is a frozen dimension, we need to test whether the assignment of constants to categories satisfies the constraints. Only if all dimensions that have been found for the query are frozen dimensions, we call the algorithm 3 to impose all the constraints in the query, so that it can be posed against the relational sources.

The ID-rewrite algorithm 3 compiles constraints, defined for the schema (FDs and KDs) by performing the following two procedures:

Algorithm 3: ID-rewrite(Ψ,Σ_I, q)

```
input  : relational schema Ψ, inclusion dependencies Σ_I,
         union of conjunctive queries Q
output: perfect rewriting of Q
Q' := Q;
repeat Q_aux := Q';
for each q ⊆ Q_aux do
    if q_i ⊆ fact f then
        then replicate f for each dimension d
    end
    for each d in dimension d do
        if q_i ⊆ d then
            then for each g ⊆ body(q) do
                for each I ⊆ Σ_I do
                    if I is applicable to g then
                        Q' := Q' U q[g/gr(g, I)]
                    end
                end
            end
            for each {q_1; q_2} ⊆ body(q) do
                if q_1 and q_2 then
                    unify Q' := Q' ∪ {τ(reduce(q, q_1, q_2))};
                end
            end
        end
    end
end
until Q_aux = Q';
return Q'
```

1. If there exists a query such that *body(q)* contains two atoms g_1 and g_2 that can be unified, then the algorithm computes the new query which is obtained from q by replacing g_1 and g_2 with U(g_1, g_2). Such a query is then added to Q.

2. If there exists an inclusion dependency IDs and the query q contains an atom g that is on the right hand side in the dependency, then the algorithm adds the query obtained from q to Q by replacing g with the atom that is on the left hand side. This step adds new conjunctions obtained by applying inclusion dependencies as rewriting rules (applied from right to left).

Termination of the algorithm is immediately implied by the fact that the number of conjunctions that can be generated by the algorithm is finite. Since the maximum length of a generated conjunction is equal to the maximum length of a conjunction in the body of the initial query Q the algorithm terminates.

4.4 Query answering complexity in the presence of IDs

In the following section we prove the separability of the conceptual dependencies and the complexity for the query answering algorithm.

It is worth noting that the above algorithm has been obtained under the assumption that the chase does not fail. In this case, we can be sure that the query rewriting algorithm does the conceptual dependencies violation repair in the same way as the chase does. Thus, in the case, when the chase fails the query is not FO-rewritable and the IDs and KDs are not separable.

To define query answering complexity, we need to mention one more crucial concept which has been defined by Cali in [24, 25]. In these papers, Cali gave a definition of the so-called Non-Key-Conflicting ID (NKCID). If the set of conceptual dependencies is a NKCID, then KDs and FDs dependencies are separable. Consequently, we can take into

account only FDs and avoid the harmful interaction of KDs and FDs.

Definition 4.4.1. *Consider a data integration system* $I =< G, S, M_{GS} >$, *with* $G =< \Psi, \Sigma_{IDs}, \Sigma_{KDs} >$, *an ID of a type* $r_1[A_1] \subseteq r_2[A_2]$ *is a Non-Key-Conflicting ID (NKCID) w.r.t. KDs if either:*

- *no KDs is defined on* r_2*;*
- *the* $KD\ key(r_2) = KDs$ *is in* Σ_{KDs} *and* A_2 *is not a strict superset of KDs, i.e.,* $A_2 \not\supseteq KDs$.

The author observed that the definition 4.4.1 includes a well known set of foreign key and primary key constraints. As we mentioned earlier, all the dependencies that we included in order to support multidimensional hierarchy can be converted to set of conceptual dependencies that are based on primary key and foreign key. Consequently, according to the definition 4.4.1, the added set of constraints are separable.

Theorem 4.4.1. *In DL-Lite, knowledge base, a satisfiability is* AC_0 *in the size of the ABox (data complexity) and EXPTIMME in the size of the whole knowledge base (combined complexity).*

Proof. The AC_0 in data complexity follows from the fact that CQ answering under inclusion dependencies has the same combined complexity in case of IDs+KDs [25]. We have shown above that IDs and KDs are separable, which means that $t \in ans(q, D, \Sigma)$ iff $t \in ans(q, D, \Sigma_{ID})$, where Σ_{ID} is a set of inclusion dependencies.

The *EXPTIME* in the data complexity relies on the proof in [24]. We will show a slight variation between the relational and multidimensional model which would add extra complexity to the problem when data is not fixed (combined complexity).

First, AC_0 data complexity follows directly from *FO - reducibility*, since evaluating FO queries/formulas over a model is AC_0 in the size of the model.

Here we prove the complexity by reducing it to an alternating linear space Turing machine M on input I.

The Alternating Turing machine naturally expands the model of a nondeterministic machine. The states of the alternating Turing machine are: existential which is accepting, if **some** transitions are accepting state or a universal state which is accepting, if **every** transition is an accepting state. Similar to [13], the strategy to tackle the proof is to construct trees to simulate the chase(D,Σ) construction. It can be done by encoding computation trees of M on the input string I, by chasing D with Σ.

According to [28], a Turing Machine(TM) is an extension of a nondeterministic TM with the addition of boolean functions - $and(\wedge), or(\vee), not(\neg)$.

Definition 4.4.2. *An alternating Turing machine(ATM) is a* $M = (k, S, \Lambda, s_0)$*, where* k *is a number of work tapes: in the case where chase is finite,* k *is a constant, otherwise chase fails and query* Q *is undecidable;* $S = S_\forall \cup S_\exists \cup S_a \cup S_r$ *partitioned into universal, existential, accepting and rejecting;* $\Lambda = 1, 0, \Gamma$*, where* Γ *fresh null;* $s_0 \in S$ *is an initial state which for the chase creation is* D*;* $\Lambda : S \times \Lambda \rightarrow (S \times \Lambda \times \{-1, 0, +1\})^2$ *is the transition function, which in our case represents constraints [13]*

While constructing the chase, a tree that is being constructed represents a set of paths (constraint application paths). Each level of the tree would represent an application of one rule, a configuration C_v of the rule v of M. Each node would contain a tape that represents configuration and a tape that represents a rule that has been applied to transfer to the state of C_v. The formal definition of D, Σ as follows:

Database D. [13] Let $D = conf(c)$, where $c \in C$ is a special constant which represents the initial configuration. A Turing machine configuration consists of the current state; a finite, possibly - empty strings of tape symbols according to the number of dimensions in the database D.

Set Σ. The constraints can be easily defined using universal and existential quantifiers.

Thus, we can easily build the chase(D,Σ) by adding new configurations based on applied constraints. We can traverse the tree as all the nodes are linked. Thus, the

complexity is *EXPTIME* for the combined complexity.

□

Corollary 4.4.1. *From the Theorem 4.4.1 we can conclude that the data complexity of the query answering over the chase(I,D) is exactly the same complexity if we take only IDs into account. Thus, the data complexity of query answering is AC_0 and the combined complexity is $EXPTIME$*

4.5 Query answering complexity in the presence of IDs+KDs

It is a well-known fact that query answering under IDs+KDs is generally undecidable. We will start considering the problem of query answering under IDs and KDs by giving a definition of non-conflicting-dependencies similar to Cali [24].

Definition 4.5.1. *Consider a data integration system $I =< G, S, M_{GS} >$, with $G =< \Psi, \Sigma_{IDs}, \Sigma_{KDs} >$, an inclusion dependency $r_1[A_1] \subseteq r_2[A_2]$ is called to be a non-conflicting-dependency with respect to Σ_{KDs} if either:*

- *there is no Σ_{KDs} that is relevant to r_2*
- *if there is Σ_{KDs} of the form $key(r_2) = K$ is in KD, and $A_2 \not\supseteq K$, which means that in inclusion dependency $r_1[A_1] \subseteq r_2[A_2]$ A_2 are the participating attributes not forming a superset of attributes that are forming a set of primary key. This schema G is NKCID if all the IDs in I are non-conflicting-dependencies.*

Informally, a set of dependencies are non-conflicting-dependencies if in a IDs a set of attributes of the relation on its right hand side is in a strict superset of the attributes in a key dependency K of the relation on its left-hand side. We prove the validity by showing that, as soon as we extend the class of dependencies beyond the defined maximal set of non-conflicting-dependencies, the query answering becomes undecidable. Assume that

we have a case where $r_1[A_1] \subseteq r_2[A_2]$ and KD $key(r_2) = K$ is in Σ_{KDs}, and if to assume that A_2 is not a maximum set of attributes K, it extends it by at most one attribute of r_2. We will call such IDs 1-Key-Conflicting IDs (1KCIDs) with respect to K.

We would like to reiterate that foreign key constraints and primary key constraints belong to the well known class of non-conflicting-dependencies. This means that $t \in ans(q, D, \Sigma)$ has the same complexity as if we had to decide whether $t \in ans(q, D, \Sigma_{ID})$.

Theorem 4.5.1. *[24] The implication problem for KDs and $1KCIDs$ is undecidable.*

Proof. Our proof heavily depends on Cali's work [24] where the non-conflicting class of dependencies is defined. They proved this by reducing from the more general problem where there are FDs and IDs. Thus, we can conclude that the same is applicable to our dimensions as we assumed that a defined category type has the same characteristics as the entities. Moreover, a new defined class of constraints use foreign key and primary key dependencies which are non-conflicting.

Consider a given database schema $G =< S, \Sigma_{IDs}, \Sigma_{FDs} >$, a set of relations and inclusion and functional dependencies(an extended class of key dependencies) and a δ which is an ID. Consider a $1KCIDs$ database schema $G_1 =< S_1, \Sigma_{IDs_1}, \Sigma_{FDs_1} >$, where Σ_{IDs_1} is a set of $1KCID$ with respect to K_1, and the same dependency δ. We will show that the two problems are equivalent, i.e. $(IDs \cup FDs) \models \delta$ iff $(IDs_1 \cup KDs_1) \models \delta$. For the conflicting database schema, we have a $G_1 =< R_1, \Sigma_{IDs_1}, \Sigma_{FDs_1} >$, and we make some specifications to show that schemas are equivalent and G_1 is conflicting, which means that G is also conflicting.

The new set of relations S_1 includes all relations in S plus r_φ. Similar to S, where we have $r(A_1, ...A_n, B_1, ..., B_k)$, we add a relation r_φ of arity $|A|+1$ of the form $r(A_1, ...A_n, A_{n+1}, B_1, ..., B_k)$ to the S set of relations. For this relation, we have a FDs $\varphi : A \rightarrow B$. To make r and r_φ equal we add two IDs:

- $\gamma_1 : \quad r\varphi[A; B] \subseteq r[A; B]$
- $\gamma_2 : \quad r[A; B] \subseteq r\varphi[A; B]$

and a key dependency

$\chi: \quad key(r_\varphi) = A$

Which leads to $K \not\subseteq A$, since cardinality of K is $|A + 1|$.

Theorem 4.5.2. *For any database B_1 for G_1, we have that B_1 satisfies $\varphi, \gamma_1, \gamma_2$, if and only if B_1 satisfies χ, γ_1, γ_2 .*

Proof. The proof heavily depends on [24]. For the dimensions and its constraints, we add a new type - category type and we showed that the category type has the same characteristics as entity. We will also need to check the non-key-conflicting set of constraints for the category/fact and category/category relations.

Hierarchical constraints are defined via foreign and primary key constraints which were proven to be non-key-conflicting. Consequently, we can conclude that the non-key-conflicting property works for dimensions.

For the fact relation we have a compound key and constraints of a type $key(F) = A$ and $F(A_1) \subseteq c(A_2)$, where there will always be $A_1 \subseteq A$.

□

Theorem 4.5.3. *Given a set of non-conflicting-dependencies for the system $I =< G, S, M >$, with $G' =< R, \Sigma_{ID}, \Sigma_{KD} >$, let $I' =< G', S, M >$ with $G =< R, \Sigma_{ID}, \varnothing >$ be the system obtained from I by eliminating the KDs of G; let D be a source database for I and I'. Moreover, let q be a query of arity n over G and G', and t an n-tuple of values. We have that $t \notin ans(q, I, D)$ iff D is consistent with Σ_{KD} and $t \notin ans(q, I', D)$.*

Proof. We say that D is consistent with KDs which means that chase(D,Σ) does not fail. Q can be evaluated over (possibly infinite) chase. The chase is obtained by applying two procedures that have been described above.

" $\Rightarrow$ " Since by hypothesis $t \notin ans(q, I, D)$, there exists a model B such that B satisfies Σ_{ID} and $t \notin ans(q, I, B)$. Since $B \in chase(D, I')$, then $t \notin chase(D, I')$.

" $\Leftarrow$ " By hypothesis, D is consistent with KD and $t \notin ans(q, I', D)$. It can be shown, by induction on the number of added tuples in the construction of chase(I,D), that if I

is a non-conflicting-dependency system, then chase(I,D) satisfies Σ_{KDs}.

It follows that chase(I,D) is a representative for all databases and $ans(q, I, D) = q^{chase}(I, D)$. □

4.6 Example of query answering algorithm under GAV mapping

In this section, we will show an example of the chase algorithm based on the given data warehouse(Example 3.2.1). Then we will show the query rewriting algorithm and prove that this algorithm is a perfect rewriting algorithm which is sound and complete.

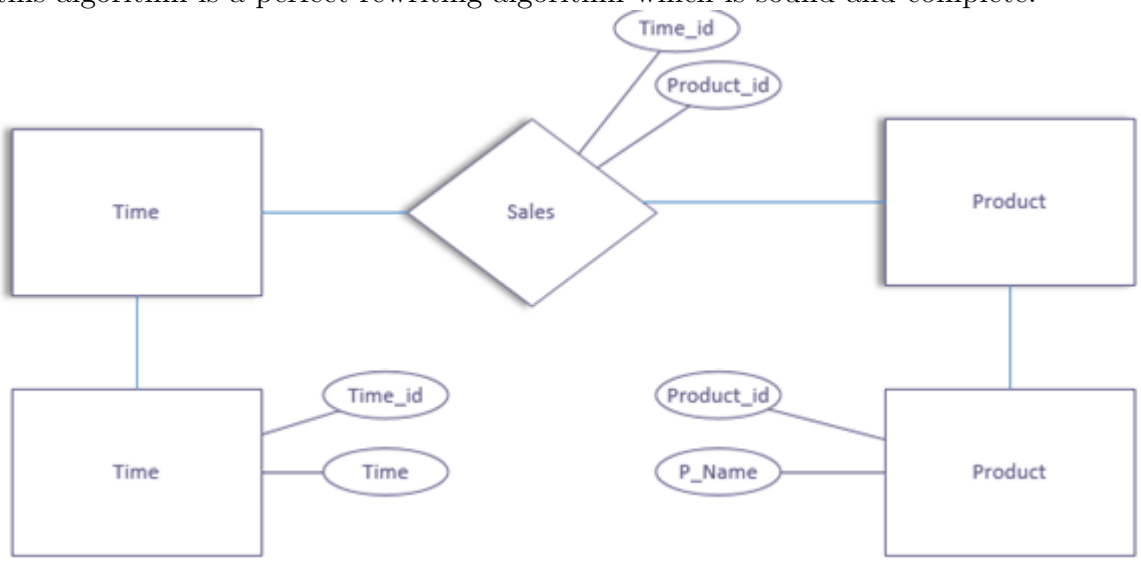


Figure 4.1: The CIM schema for the Example 3.2.1

Example 4.6.1. *A schema shown in the Figure 4.1 can be represented as a set of the following relations plus a set of key and inclusion dependencies.*

$$Sales\{Employee\_id,\ Time\_id\}$$

$$Employee\{Employee\_id,\ Name\}$$

$$Time\{Time\_id,\ Time\}$$

and the following constracts

$$key\{Sale\} = \{Employee\_id,\ Time\_id\}$$

$$key\{Employee\} = \{Employee\_id\}$$

$$key\{Time\} = \{Time\_id\}$$

$$Sales\{Employee\_id\} \subseteq Employee\{Employee\_id\}$$

$$Sales\{Time\_id\} \subseteq Time\{Time\_id\}$$

Now assume, that we are given the following database and a query posed against it.

Time_id	Time

Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread
14/05	cheese

Product_id	Product

Figure 4.2: Step 0: Initial data

A query posed against this database is

$$Q(Time\_id, Product\_id) \leftarrow Sale(Time\_id\,, Product\_id),$$
$$Product(Product\_id), Time(14/05)$$

The procedure of the chase algorithm application is:

Step 1

$Sales(Time\_id) \subseteq Time(Time\_id)$

Time_id	Time
14/05	
15/05	

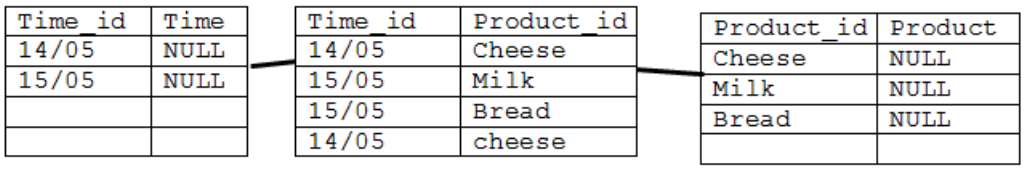
Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread
14/05	cheese

Product_id	Product

Figure 4.3: Chase: Step 1

Step 2

$Sales(Product\_id) \subseteq Product(Product\_id)$



Time_id	Time
14/05	NULL
15/05	NULL

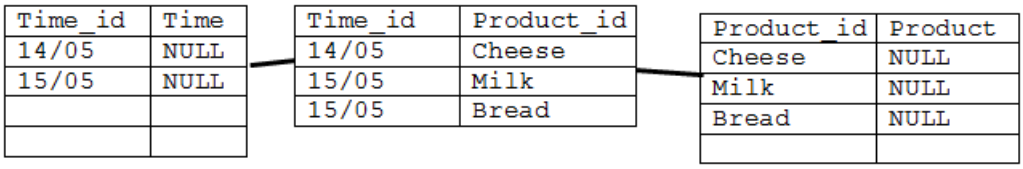
Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread
14/05	cheese

Product_id	Product
Cheese	NULL
Milk	NULL
Bread	NULL

Figure 4.4: Chase: Step 2

Step 3

$KeySale = (Time\_id,\ Product\_id)$



Time_id	Time
14/05	NULL
15/05	NULL

Time_id	Product_id
14/05	Cheese
15/05	Milk
15/05	Bread

Product_id	Product
Cheese	NULL
Milk	NULL
Bread	NULL

Figure 4.5: Chase: Step 3

The query

$Q(Time\_id, Product\_id) \leftarrow$

$Sale(Time\_id,\ Product\_id), Product(Product\_id), Time(14/05)$

This query, posed against $chase(D, \Sigma)$ *would return* $\{14/05, Cheese\}$

On the other hand, after applying the query rewriting algorithm we obtain the following query:

*Step*0

$Q(Time\_id,\ Product\_id) \leftarrow Sale(Time\_id,\ Product\_id),$

$Sale(Time\_id, Product\_id), Product(Product\_id), Time(14/05)$

$Step1$

$Sales(Time\_id) \subseteq Time(Time\_id)$

$Q(Time\_id,\ Product\_id) \leftarrow Sale(Time\_id,\ Product\_id),$

$Sale(Time\_id, Product\_id), Product(Product\_id), Sale(14/05, \_\_)$

$Step2$

$Sales(Product\_id) \subseteq Product(Product\_id)$

$Q(Time\_id,\ Product\_id), \leftarrow Sale(Time\_id,\ Product\_id),$

$Sale(Time\_id, Product\_id)Sale(\_\_, Product\_id), Sale(14/05, \_\_)$

$Step3$

$Unify$

$Q(Time\_id,\ Product\_id) \leftarrow Sale(Time\_id, Product\_id), Sale(Time\_id, \_\_,)$

$Sale(14/05, \_\_)$

$Step4$

$Unify$

$Q(Time\_id, Product\_id) \leftarrow Sale(\_\_, Product\_id), Sale(14/05, \_\_)$

$Step5$

$Unify$

$Q(Time\_id, Product\_id) \leftarrow Sale(14/05, Product\_id)$

The query

$Q(Time\_id, Product\_id) \leftarrow Sale(14/05, Product\_id)$

That was returned by rewriting the algorithm posed against D would return $\{14/05, Cheese\}$

Lemma 4.6.1. (Loss of soundness) *:*

Consider the set Σ *of TGDs given in Example 4.6.1, and also the CQ*

$$Q(Time\_id, Product\_id) \leftarrow Sale(Time\_id, Product\_id), \\ Product(Product\_id), Time(14/05)$$

Dropping the $q[g/gr(g, I)]$ *would lead to the CQ* q' *:*

$$Q(Time\_id, Product\_id) \leftarrow Sale(14/05, Product\_id)$$

This query will return NULL as the original query q does not map to chase(D, Σ*), and thus,* $D \cup \Sigma \not\models q$*. Therefore, any rewriting containing* q' *is not a sound rewriting of q given* Σ*.*

Dropping condition (2) would lead to

$$Q(Time\_id, Product\_id) \leftarrow Sale(Time\_id, Product\_id), \\ Sale(Time\_id, Product\_id), Sale(\_\_, Product\_id), Sale(14/05, \_\_)$$

However, during the construction of the chase(D, Σ) it is not possible to get atoms $Sale(Time\_id, Product\_id)$ and $Sale(14/05, Product\_id)$, where $Time\_id$ would differ from 14/05 since the combination $\{Time\_id, Product\_id\}$ is unique. This implies that there is no homomorphism that maps q to chase(D, Σ), and hence $D \cup \Sigma \models q$. Therefore, any rewriting containing q' is unsound. This implies that there is no homomorphism that maps q to chase(D, Σ), and hence $D \cup \Sigma \not\models q'$. Therefore, any rewriting containing q' is unsound.

Lemma 4.6.2. (Loss of completeness)*: Consider the set* Σ *of TGDs given in Example 4.6.1, aw well as the CQ*

$$Q(Time\_id, Product\_id) \leftarrow Sale(Time\_id, Product\_id), \\ Product(Product\_id), Time(14/05)$$

Without rewriting the algorithm it is impossible to obtain a resulting query which is

$Q(Time\_id, Product\_id) \leftarrow Sale(14/05, Product\_id)$

Now, consider the database that is given in the Example 4.6.1. Clearly, $chase(D, \Sigma) \models q$, or, equivalently, $D \cup \Sigma \models q$. However, the rewritten query is not entailed by the given database D which implies that it is not complete.

Corollary 4.6.1. *Since we proved that the* $D \models DimBUILD(D, \Sigma, Q)$ *iff* $D \cup \sigma \models q$, *we can claim that the algorithm is the perfect rewriting technique. Thus, the query is first-order rewritable and it is well-known that evaluating first-order queries is in the highly tractable class* AC_0 *in data complexity.*

4.7 Summarizing results

In the following we summarize our results in a small table and discuss the changes that appeared after we took into account summarizability.

Complexity summary			
Mapping	CD	Data complexity	Combined complexity
GAV	$ID + KD + dim$	AC_0	$EXPTIME$
	$ID + KD$	$undec$	$undec$
	$ID + dim$	AC_0	$EXPTIME$

We first showed that the conceptual dependencies IDs+KDs that are defined in [20] can be applied to define semantics in our CIM. However, we then showed examples, where we would need a more expressive semantics to capture the hierarchical structure of the dimensions. Consequently, we came to a conclusion that we would need a more specific set of constraints to make the problem of query answering decidable. If to consider a simple set of conceptual dependencies IDs+KDs the query answering problem is decidable. We proved that the set of conceptual dependencies IDs+KDs is separable, so we can consider only IDs avoiding harmful interaction between IDs and KDs. Thus, the query answering complexity is in highly tractable class of AC_0. However, the combined complexity raised

significantly to $EXPTIME$. Though, it makes sense since the complexity of Datalog is $EXPTIME$ and we cannot have the query answering complexity grater than that.

4.8 Conclusion

In this section we presented the query answering algorithm that imposes conceptual dependencies in a conjunctive query so it can be imposed against relational sources directly. We showed that, although the hierarchical multidimensionality seems to have dramatic impact on the complexity of the query answering algorithm, in practice it moderately increased the complexity. In the following chapter we will tackle the same problem but in this case we will consider local-as-view approach of mapping.

Chapter 5

Query answering in the LAV approach

In this section we prove the complexity of the query answering algorithm in the CIM with respect to local-as-view approach of mapping. We will show why we cannot use the algorithm we defined for the GAV. We will give a definition of minimal dimension and present an algorithm for query answering w.r.t LAV.

5.1 Local-as-a-view approach of mapping

In a conceptual integration model $I =< G, S, M >$ based on the local-as-a-view approach, the mapping M assigns a view, a query q^G over G to each relation s of the source schema S. In other words, in LAV, the results are constrained by the objects in sources, while the global conceptual schema may contain richer information. Thus, in the LAV systems, it may be necessary to deal with inconsistent answers. As we will see from the example, if some dimensions are missing in the sources, the query answers are aggregated over these dimensions which leads to the inconsistent data in a global database.

Similarly to the GAV approach, we consider sound view which means that a source s provides any subset of the tuples satisfying the corresponding view q^G.

Let us study the question of query decidability in LAV, where views are considered sound. The global schema is expressed as a set of relations plus certain constraints, and the queries are expressed in the Datalog sentences. On the contrary, in the following we will define the constraint that will help the query decidability. In particular, along with inclusion dependencies and equality-generating dependencies, we will define a constraint for dimensions - that sets a minimal set of data that will suffice for query decidability.

5.2 Example of query answering algorithm under LAV mapping

The following example rephrases an example given in [51].

Example 5.2.1. *Consider a data integration system I with global relational schema G containing three dimensions: employee, time, product and a fact relation.*

$$Sale = \{Employee\_id, Time\_id; Product\_id\}$$

$$Employee = \{Employee\_id, Employee\_Name\}$$

$$Time = \{Time\_id, Time\}$$

$$Product = Product\{Product\_id, Brand\_id\}$$

Figure 5.2.1 is a conceptual schema, a shrunken version of a global Example (will add to the report my very first example later).

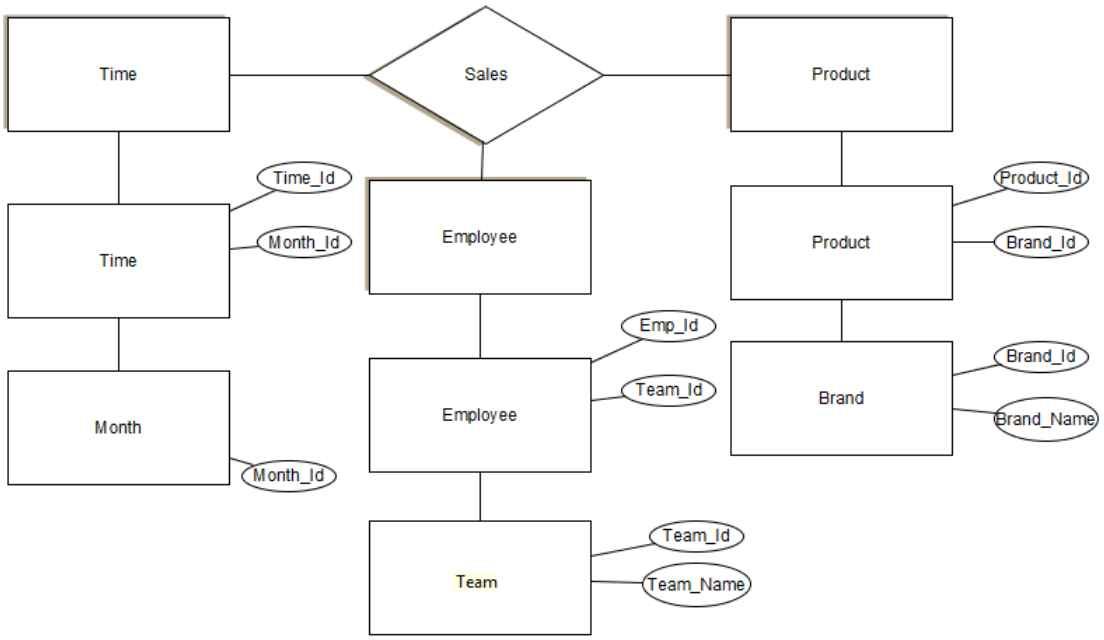


Figure 5.1: The CIM schema for the example

Consider three sources with different combinations of two dimensions from Employee, Time, Product.

Source 1

$$Sale = \{Employee\_id, Time\_id\}$$

$$Employee = \{Employee\_i, Employee\_Name\}$$

$$Time = \{Time\_id, Time\}$$

Source 2

$$Sale = \{Time\_id, Product\_id\}$$

$$Time = \{Time\_id, Time\}$$

$$Product = Product\{Product\_i, Brand\_id\}$$

Source 3

$$Sale = \{Employee\_id, Product\_id\}$$

$$Employee = \{Employee\_id, Employee\_Name\}$$

$$Product = Product\{Product\_id, Brand\_id\}$$

and consider a source database presented on the Figures(5.2,5.3,5.4)

Time_Id	Time
01/08/15	Aug, 1st
02/08/15	Aug, 2nd

Time_Id	Emp_Id	Amount
01/08/15	John	3
02/08/15	Kelly	5

Emp_Id	Employee Name
John	John Brown
Kelly	Kelly Smith

Figure 5.2: The CIM schema for the example

Time_Id	Time
01/08/15	Aug, 1st
02/08/15	Aug, 2nd

Time_Id	Product_Id	Amount
01/08/15	Cheese	3
02/08/15	Coke	5

Product_Id	Product Name
Cheese	Holland cheese
Coke	Pepsi coke

Figure 5.3: The CIM schema for the example

Product_Id	Product
Cheese	Holland cheese
Coke	Pepsi coke

Product_Id	Emp_Id	Amount
Cheese	John	3
Coke	Kelly	5

Emp_Id	Employee Name
John	John Brown
Kelly	Kelly Smith

Figure 5.4: The CIM schema for the example

If sources are sound, we obtain the data aggregated by different dimensions, thus, it cannot be integrated. On the other hand, if both sources are exact, we can conclude that the data is semantically consistent.

The formalization of data integration presented in the previous sections is based on a first order logic interpretation, and, therefore, is not able to cope with inconsistencies between sources. Indeed, if the data in the system $I =< G, S, M >$ does not satisfy the integrity constraints of G, then no global database exists for I, and the query answering becomes meaningless. This is the situation that occurs when data in the sources are

mutually inconsistent. Generally speaking, the problem is to define a minimal set of dimensions over which query can be processed.

Now we are ready to define a minimal set of dimensions that are in the sources to be integrated.

Definition 5.2.1. *Given a data integration system* $I =< G, S, M >$, *a* $G =< C, \Sigma_{ID}, \Sigma_{KD} >$, *a minimal set of dimensional relations (*dim_{min}*) is a*

$$dim_{min} = \{dim | dim \in S_i, where \quad S_i \in S\}$$

Finally, we give the semantics of queries.

Definition 5.2.2. *[24] Given a database instance* D *and sources* S, *we call answers to a query* q *of arity* n *with respect to* G, *to be a set* $t \in ans(q, G, D)$ *such that* $ans(q, G, D) = \{< c_1, ..., c_n > — for\ each\ source\ B \in sem(G, D), < c_1, ..., c_n > \in q_i^B\}$, *where* q_i^B *denotes the result of evaluating* q *over a source database* B_i. q *of the form* $\exists y_1.conj_1(\overrightarrow{x_1}, \overrightarrow{y_1}) \vee ...conj_n(\overrightarrow{x_n}, \overrightarrow{y_n})$ *such that* $conj_i(\overrightarrow{x_i}, \overrightarrow{y_i}) \in dim_{min}$ *or* $conj_i(\overrightarrow{x_i}, \overrightarrow{y_i}) \in fact$. *We recall that* q^B *is the set of n-tuples of values of* $\Gamma < c_1, ..., c_n >$, *such that, when substituting each* c_i *for the formula* $\exists y_1.conj_1(\overrightarrow{x_1}, \overrightarrow{y_1}) \vee ...conj_n(\overrightarrow{x_n}, \overrightarrow{y_n})$ *is evaluated to true in each source* B.

Now we are ready to present an algorithm to define whether a tuple t is a consistent answer to a query q. A tuple t is a consistent answer over the sources with respect to S and D, i.e., $t \in ans_{LAV}(q, G, D)$, iff for every $S' \subseteq S$ $ans(q, S', D)$ returns true for each such source database S'. More specifically, we define the algorithm below.

From the algorithm we can see that condition (1) checks that S' is a maximal subset of S consistent with KD; Condition (2) checks that a new added tuple is also consistent with KD and has the minimal number of dimensions; This implies the existence of a source database B such that $B \cap D = D'$. Then, condition (3) verifies that $t \in q^B$.

Algorithm 4: $ID\text{-}rewrite_{LAV}$

input : database schema $G =< C, \Sigma_{ID}, \Sigma_{KD} >$, database instance D, query q of arity n over G, source schema S;
output: perfect rewriting of Q;
If there exists $S' \subseteq S$ such that

- S' is consistent with KDs;
- **for each** $s \in S$
  - $S' \cup s$ is not consistent with KDs;
  - *or* $< c_1, ..., c_n >$ in $c_i \notin dim_{min}$
- $ans(q, G, D)$ returns *false*

then return *null*
else return $DimBUILT(S', \Sigma_{ID} \cup \Sigma_{KD}, q)$

5.3 Query answering complexity in the presence of IDs + KDs

Theorem 5.3.1. *Consider a data integration system* $I =< G, S, M_{GS} >$, *where* $G =< C, \Sigma_{ID}, \Sigma_{KD} >$ *be a global schema,* D *be a database instance for* I, q *be a query of arity* n *over* G, *and* t *be a* n*-tuple of values of* Γ. *The problem of establishing whether* $t \in ans_{LAV}(q, I, D)$ *is coNP-complete with respect to data complexity.*

Proof. The proof heavily relies on [24]. In [24] the authors used a reduction of the 3-colorability problem to their problem. Consider a graph $G = (V, E)$ with a set of vertices V and edges E. We define a data warehouse schema $DB =< S, \emptyset, KD >$ where S consists of the one fact table and two dimensions *edge* and *col*, and KD contains the dependency $key(fact) = \{1, 2\}$. The instance D is defined as follows:

$$D = \{col(c; i) | i \in 1, 2, 3 \text{ and } c \in V\} \cup \{edge(e; x; y) | e \in E \text{ and } x, y \subset V\} \cup \{fact(e; c)\}$$

Finally, we define the query

$$q(X) \leftarrow fact(e, X); edge(e, X, Y); col(Y; "1")$$

There are two cases to consider:

- If we consider that sources do not have one of the dimensions and data integration will be meaningless.
- If we consider that sources have the minimal set of dimensions(in our case sources have either one or both dimensions) then we can rely on the proof that has been presented in [24] and thus, the complexity is $co - NP$. As we check the KD consistency for the D', the proved complexity holds for KD and $KD+ID$ together.

□

Theorem 5.3.2. *[24] Consider a data integration system $I =< G, S, M_{GS} >$, where $G =< C, \Sigma_{ID}, \Sigma_{KD} >$ be a global schema, D be a database instance for I, q be a query of arity n over G, and t be a n-tuple of values of Γ. The problem of establishing whether $t \in ans(q, I, D)$ is $EXPTIME$ complete with respect to combined complexity.*

Proof. Hardness follows from the algorithm and from the fact that, when D is consistent with K and $dim \in dim_{min}$ we will have to consider $|D|^x$ of all the possible combinations to determine whether $t \in ans(q, I, D)$. Indeed, it is easy to see that conditions (1), (2), and (3) of the Algorithm 4 can be verified in exponential time, and furthermore *EXPTIME*

□

5.4 Summarizing results

In the following section we will summarize the results regarding the query answering complexity with respect of LAV approach of mapping. In this chapter we considered

LAV approach of mapping, which means that source schemas might be more aggregated than the global conceptual schema. In this chapter we defined the notion of minimal dimensions and presented the query answering algorithm.

Complexity summary			
Mapping	CD	Data complexity	Combined complexity
LAV	$ID + KD + dim$	$co - NP$	$EXPTIME$
	$ID + KD$	$undec$	$undec$
	$KD + dim$	$co - NP$	$EXPTIME$

From the table we can see that IDs+KDs constraints, that are defined by Dr. Cali are not enough to capture the semantics of the CIM with local-as-view approach of mapping. Thus, the query answering complexity is undecidable. Interestingly, the combined complexity of the query answering algorithm with respect to the combined set of IDs+IKs+dim is the same as in the case of GAV approach. It is not a surprise, since the complexity of Datalog logic is $EXPTIME$ and we can not have query answering complexity greater than that.

5.5 Conclusion

In this chapter we showed how queries can be answered with respect to the local-as-view approach of mapping. We showed why we cannot use the algorithm we defined for the GAV. We defined a notion of minimal dimension to make the query answering decidable. We proved the complexity of query answering algorithm in the CIM with respect to local-as-view approach of mapping. The combined complexity of the query answering algorithm with respect to the combined set of IDs+IKs+dim is $EXPTIME$ while the data complexity is $co - NP$.

Chapter 6

Conclusion

In this chapter we draw several conclusions based on our defined semantics of the conceptual integration model and mapping language that made up the scope of this work. Furthermore, we provide the results and implications.

The Conceptual Integration Modelling Framework has been developed to offer a user-oriented interaction with data warehouses. However, a formal foundation in terms of semantics of involved conceptual schemas had not yet been defined. Moreover, the decidability of conceptual queries was yet to be addressed.

In this thesis, we extended formal foundations that have been given in [20, 25] for non-dimensional conceptual schemas to capture multidimensional conceptual schemas. It turned out that, in order to do so, we faced the problem of capturing the fundamental property of summarizability that is associated with multidimensional hierarchical schemas. We first associated conceptual schemas with a set of relations augmented with inclusion dependencies and key dependencies. Then we showed that the Equality Generating Dependencies and Tuple Generating Dependencies are not enough to guarantee dimensional summarizability. We enriched the existing semantics studied in [20] to capture summarizability.

With respect to the addressed problem of the complexity of answering conceptual queries posed against conceptual schemas, we found out that the additional constraints

increased the complexity from *PSPACE* (which was proven in [20, 25]) to *EXPTIME*.

Studying the query answering was be done by compiling user queries CQ using DL-Lite ontologies [39] into a new CQ' with imposed conceptual dependencies. We used the language of Description Logic ontologies because it is an appropriate logical query language for checking satisfiability of a knowledge base and answering complex queries.

We presented query answering algorithms that add conceptual dependencies in a conjunctive query that can be imposed against data warehouses directly. We defined the algorithms with respect to two kinds of mappings: one where elements from a global conceptual schema are mapped to a view over the sources (in our case, relational data warehouses) and, an other where sources contain more aggregated information than the conceptual schema. We showed that, although the hierarchical multidimensionality seems to have dramatic impact on the complexity of the query answering algorithm, in practice it moderately increased the complexity.

Bibliography

[1] Cognos web site. http://www.cognos-bi.info/cognos8.html, 2007. [Online; accessed 28-June-2015].

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

[3] Atul Adya, José A Blakeley, Sergey Melnik, and S Muralidhar. Anatomy of the ado. net entity framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 877–888. ACM, 2007.

[4] Malaki Aida, Bertossi Leopoldo, and Rizzolo Flavio. Multidimensional contexts for data quality assessment. *CEUR Workshop Proceedings*, 14, 2012.

[5] Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z Pan. *The Semantic Web: Research and Applications: 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29–June 2, 2011. Proceedings*, volume 6644. Springer, 2011.

[6] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 68–79. ACM, 1999.

[7] Yigal Arens, Chin Y Chee, Chun-Nan Hsu, and Craig A Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(02):127–158, 1993.

[8] Mohammad Ariyan. *A multidimensional data model with subcategories for expressing and reparing summarizability*. PhD thesis, Carleton university Ottawa, 2014.

[9] Sina Ariyan and Leopoldo Bertossi. A multidimensional data model with subcategories for flexibly capturing summarizability. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, page 6. ACM, 2013.

[10] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The dl-lite family and relations. *Journal of artificial intelligence research*, 36(1):1–69, 2009.

[11] Daniele Barone, Liam Peyton, Flavio Rizzolo, Daniel Amyot, and John Mylopoulos. Towards model-based support for managing organizational transformation. In *E-Technologies: Transformation in a Connected World*, pages 17–31. Springer, 2011.

[12] Daniele Barone, Eric Yu, Jihyun Won, Lei Jiang, and John Mylopoulos. Enterprise modeling for business intelligence. In *The practice of enterprise modeling*, pages 31–45. Springer, 2010.

[13] Pierre Bourhis, Michael Morak, and Andréas Pieris. Acyclic query answering under guarded disjunctive existential rules and consequences to dls. In *Informal Proceedings of the 27th International Workshop on Description Logics*, 2014.

[14] Boury-Brisset and Anne-Claire. Managing semantic big data for intelligence. In *STIDS*, pages 41–47, 2013.

[15] Francois Bry. Query answering in information systems with integrity constraints. In *Integrity and Internal Control in Information Systems*, pages 113–130. Springer, 1997.

[16] Doug Burdick, AnHai Doan, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Olap over imprecise data with domain constraints. In *Proceedings of the 33rd in-*

ternational conference on Very large data bases, pages 39–50. VLDB Endowment, 2007.

[17] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Rewrite and conquer: Dealing with integrity constraints in data integration. In *Seminal Contributions to Information Systems Engineering*, pages 353–359. Springer, 2013.

[18] Andrea Cali, Diego Calvanese, and Maurizio Lenzerini Giuseppe De Giacomo. Data integration under integrity constraints. *In Proc. of CAiSE*, 18, 2002.

[19] Andrea Cali, Georg Gottlob, and Andreas Pieris. Tractable query answering over conceptual schemata. *Proc. of the 28th Intl Conf. on Conceptual Modeling (ER)*, 15, 2009.

[20] Andrea Cali, Georg Gottlob, and Andreas Pieris. Query answering under expressive entity-relationship schemata. *In Proc. of ER*, 15:347–361, 2010.

[21] Andrea Calí, Georg Gottlob, and Andreas Pieris. Ontological query answering under expressive entity-relationship schemata. *Information Systems*, 37(4):320–335, 2012.

[22] Andrea Cali, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.

[23] Andrea Cali, Georg Gottlob1, and Andreas Pieris. Querying conceptual schemata with expressive equality constraints. *ER'11*, 13, 2011.

[24] Andrea Cali, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. *PODS '03*, 11, 2003.

[25] Andrea Cali, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. *IJCAI*, 6, 2003.

[26] Andrea Cali and Davide Martinenghi. Querying incomplete data over extended er schemata. *Theory and Practice of Logic Programming*, 38, 2010.

[27] De Giacomo G. Calvanese, D. and M Lenzerini. On the decidability of query containment under constraints. *Proc. of PODS'98*, 9, 1998.

[28] Ashok K Chandra and Larry J Stockmeyer. Alternation. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 98–108. IEEE, 1976.

[29] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The tsimmis project: Integration of heterogenous information sources. 1994.

[30] Theodore Dalamagas, Dimitri Theodoratos, Antonis Koufopoulos, and I-Ting Liu. Semantic integration of tree-structured data using dimension graphs. In *Journal on Data Semantics IV*, pages 250–279. Springer, 2005.

[31] Calvanese Diego, Lembo Domenico De Giacomo, Giuseppe, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *KR*, 6:260–270, 2006.

[32] Domenico Lembo Maurizio Lenzerini Riccardo Rosati Diego Calvanese, G. De Giacomo. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *J. of automated resoning*, page 56, 2007.

[33] D.Shakya, C. Chen, F. Nargesian, F. Rizzo, Y. Lou, M. Smith, I. Kiringa, and R. Pottinger. The conceptual integration modeling framework: Abstracting from the multidimensional model. *Demo to the 2011 BIN Workshop*, 14, 2011.

[34] E. Zimányi. E. Malinowski. Advanced data warehouse design: From conventional to spatial and temporal applications.

[35] Ronald Fagin, Laura M Haas, Mauricio Hernández, Renée J Miller, Lucian Popa, and Yannis Velegrakis. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*, pages 198–236. Springer, 2009.

[36] Ronald Fagin, Phokion G Kolaitis, Renée J Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[37] Dominique Ferrand, Daniel Amyot, and Carlos Villar Corrales. Towards a business intelligence framework for healthcare safety. *Journal of Internet Banking and Commerce*, 15(3):1–9, 2010.

[38] Michael Franklin, Alon Halevy, and David Maier. From databases to dataspaces: a new abstraction for information management. *ACM Sigmod Record*, 34(4):27–33, 2005.

[39] Georg Gottlob, Thomas Lukasiewicz, et al. A general datalog- based framework for tractable query answering over ontologies. 2009.

[40] Alon Y Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[41] Richard Hull. Managing semantic heterogeneity in databases: a theoretical prospective. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 51–61. ACM, 1997.

[42] Carlos A Hurtado, Claudio Gutierrez, and Alberto O Mendelzon. Capturing summarizability with integrity constraints in olap. *ACM Transactions on Database Systems (TODS)*, 30(3):854–886, 2005.

[43] Carlos A Hurtado and Alberto O Mendelzon. Olap dimension constraints. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 169–179. ACM, 2002.

[44] Carlos A Hurtado, Alberto O Mendelzon, and Alejandro A Vaisman. Updating olap dimensions. In *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*, pages 60–66. ACM, 1999.

[45] Zachary G Ives. *Efficient query processing for data integration*. PhD thesis, University of Washington, 2002.

[46] Leonid Kalinichenko and Sergey Stupnikov. Synthesis of the canonical models for database integration preserving semantics of the value inventive data models. In *Advances in Databases and Information Systems*, pages 223–239. Springer, 2012.

[47] Benedikt Kämpgen, Steffen Stadtmüller, and Andreas Harth. Querying the global cube: Integration of multidimensional datasets from the web. In *Knowledge Engineering and Knowledge Management*, pages 250–265. Springer, 2014.

[48] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley; 2 edition (April 26 2002, 2002.

[49] Ruzzi Lembo. Consistent query answering over description logic ontologies. 8, 2007.

[50] Hans-J Lenz and Arie Shoshani. Summarizability in olap and statistical data bases. In *Scientific and Statistical Database Management, 1997. Proceedings., Ninth International Conference on*, pages 132–143. IEEE, 1997.

[51] Maurizio Lenzerini. Data integration: A theoretical perspective. *PODS '02*, 13, 2002.

[52] Jose-Norberto Mazón, Jens Lechtenbörger, and Juan Trujillo. Solving summarizability problems in fact-dimension relationships for multidimensional models. In *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP*, pages 57–64. ACM, 2008.

[53] Mostafa Milani, Leopoldo Bertossi, and Sina Ariyan. Extending contexts with ontologies for multidimensional data quality assessment. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 242–247. IEEE, 2014.

[54] Dave Minter and Jeff Linwood. *Beginning Hibernate: From Novice to Professional*. Dreamtech Press, 2006.

[55] Naoual Mouhni and Abderrafiaa EL KALAY. A critical overview of existing query processing systems over heterogeneous data sources. *Journal of Theoretical & Applied Information Technology*, 60(2), 2014.

[56] Fatemeh Nargesian. *Bridging decision applications and multidimensional databases*. PhD thesis, University of Ottawa, 2011.

[57] Fatemeh Nargesiana, Flavio Rizzoloa, Iluju Kiringa, and Rachel Pottingerb. Supporting conceptual and storage multidimensional integration for data warehouse creation. *Technical report, SITE, University of Ottawa*, 24, 2011.

[58] Tapio Niemi, Marko Niinimäki, Peter Thanisch, and Jyrki Nummenmaa. Detecting summarizability in olap. *Data & Knowledge Engineering*, 89:1–20, 2014.

[59] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. In *International semantic web conference*, volume 4273, pages 30–43. Springer, 2006.

[60] Pottinger and Rachel Amanda. *Processing queries and merging schemas in support of data integration*. PhD thesis, Citeseer, 2004.

[61] MP Reddy, Bandreddi E Prasad, PG Reddy, and Amar Gupta. A methodology for integration of heterogeneous databases. *Knowledge and Data Engineering, IEEE Transactions on*, 6(6):920–933, 1994.

[62] Flavio Rizzolo, Iluju Kiringa, Rachel Pottinger, and Kwok Wong. The conceptual integration modeling framework: Abstracting from the multidimensional model. *Technical report, SITE, University of Ottawa*, 14, 2010.

[63] Rosati and Riccardo. Query rewriting under extensional constraints in dl-lite. In *25th International Workshop on Description Logics*, page 323. Citeseer, 2012.

[64] Johnson David S and Klug Anthony. Testing containment of conjunctive queries under functional and inclusion dependencies. In *Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems*, pages 164–169. ACM, 1982.

[65] Dibesh Shakya. The model manager: A multidimensional conceptual modeling tool in the cim framework.

[66] Amit P Sheth and James A Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 22(3):183–236, 1990.

[67] Bernhard Thalheim. The enhanced entity-relationship model. In *Handbook of Conceptual Modeling*, pages 165–206. Springer, 2011.

[68] Nectaria Tryfona, Frank Busborg, and Jens G Borch Christiansen. starer: a conceptual model for data warehouse design. In *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*, pages 3–8. ACM, 1999.

[69] Moshe Y. Vardi. The complexity of relational query languages. *STOC '82*, 9, 1982.

[70] wikipedia. wiki Data warehouse. https://en.wikipedia.org/wiki/Data_warehouse, 2008. [Online; accessed 26-June-2015].

[71] Wikipedia. Abox — Wikipedia, the free encyclopedia, 2015. [Online; accessed 29-Oct-2015].

[72] Wikipedia. Datalog — Wikipedia, the free encyclopedia, 2015. [Online; accessed 29-Oct-2015].

[73] Wikipedia. Tbox — Wikipedia, the free encyclopedia, 2015. [Online; accessed 29-Oct-2015].

[74] Haifei Zhang. A query driven method of mapping from global ontology to local ontology in ontology-based data integration. *Journal of Software*, 9(3):738–742, 2014.

[75] Patrick Ziegler and Klaus R Dittrich. Data integration—problems, approaches, and perspectives. In *Conceptual Modelling in Information Systems Engineering*, pages 39–58. Springer, 2007.