Equivalence of OLAP Dimension Schemas

Carlos A. Hurtado and Claudio Gutiérrez
Department of Computer Science
University of Chile
{churtado,cgutierr}@dcc.uchile.cl
Blanco Encalada 2120, Santiago, Chile , C.P. 6511224

Paper ID: 156

1 Introduction

OLAP dimensions are data hierarchies that populate data warehouses. These entities are hierarchically organized information that define the perspective upon which the data is viewed. As an example, in a data warehouse we may have dimensions describing products, stores and time, which may be used to visualize the facts generated by a sales process. Figure 1 depicts a dimension that models financial services offered by a bank: accounts, credit cards and loans. The products are classified through the hierarchy path $Product$-$ProdType$-$ProdCategory$-All. Some types of products, like personal loans and some sorts of accounts, are handled by branches, whereas others, like mortgage and corporate loans, are handled by departments. The products handled by branches are also classified according to the category $BranchProdType$. There is a manager in charge of each branch and department. Finally, it happens that all departments handle products in only one category. On the left hand side of Figure 1, there is a graph called *hierarchy schema* which models the structure of the dimension. The vertices of this graph are called categories. On the right hand side, there is another graph, called *hierarchy domain*, whose vertices, called members, are grouped by categories and ordered by a child/parent relation. For example, in the dimension at hand, we may say that member $p1$ belongs to the category $Product$ and $p1$ has $d1$ as a parent in the category $Department$.

Dimension Schema A *dimension schema* is an abstract model of a dimension commonly used to support summarizability reasoning in OLAP applications [HM01], that is, to test whether aggregate views defined for some categories can be correctly derived from a set of precomputed views defined for other categories. In previous work [HM02] we have introduced semantically reach dimension schemas to support this inference task. A dimension schema, being an abstract representation of a dimension, represents the set of possible dimensions that conforms to it. This set reflects the *information capacity* of the schema. Thus when we perform reasoning on the schema, we infer properties of all the dimensions in the set.

Dimension schemas are modeled as a hierarchy schema (i.e. the structure of the dimension) along with a set of integrity constraints, called *dimension constraints*. The hierarchy schema is a directed acyclic graph whose vertices are the categories, and whose edges capture the child/parent relation among the members. The constraints are used to place further restrictions to let the schema capture more precisely different sets of dimensions. The most basic constraints are statements about paths arising in the child/parent relation. For example, we may require that all the products handled by some branch are not handled by departments, and viceversa. This is stated by the constraint saying that for each product, it can have ancestors in either the path $\langle Product, Branch\rangle$ or the path $\langle Product, Department\rangle$, but not in both. Other constraints may express that the ancestor of some members is another particular member. For example we may state "the manager of the Asia branch is Mr. Huang" as "$\langle Branch = Asia\rangle$ implies $\langle Branch, .., Manager = Huang\rangle$". The expressions in brackets are atomic statements (called *atoms*). It turns out that Boolean combinations of atoms are needed to support summarizability reasoning [HM02].

Simple forms of these constraints characterize typical classes of OLAP schemas. In this sense, this extended class of schemas with constraints subsumes other well known classes in OLAP.

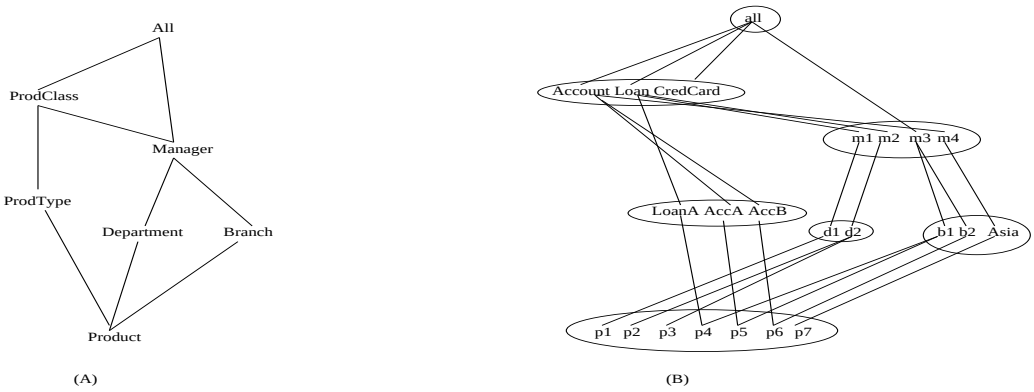


Figure 1: The dimension Product: (A) hierarchy schema; (B) child/parent relation.

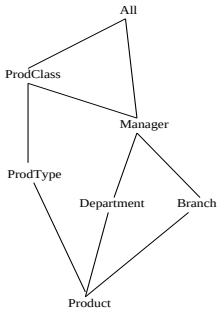
Problem Statement Similarly to the case of general database schemas, two dimension schemas could be compared with respect to their information capacity. Schemas with the same information capacity can be used to simulate each other. Having different equivalent schemas give users flexibility to choose among several options the best suited for the application at hand. In a typical modeling scenario the user starts with some schema and proceed to restructure it. In the context of OLAP, it is very important that the restructuring process preserves schema equivalence because the schema is more useful for reasoning about data than it is just as a container of data. So we would like to keep the information on the schema as precise as possible to capture the set of instances as tight as possible.

Formal notions of schema equivalence are needed to sit restructuring mechanisms and schema design techniques on solid grounds. For example, Miller et al. [MIR94] argue that the restrestructuringcess may be addressed following two different strategies: (i) build a desired schema and then test whether it is equivalent to the original schema; (ii) use a set of primitives to transform the original schema into a desired schema. In both approaches, we need to define under which conditions two dimension schemas are equivalent. In the first approach we need algorithms for the equivalence test. The second approach requires a set of well defined dimension transformations. The central desirable properties of such a set, soundness and completeness [Alb00], depend on the notion of schema equivalence used as well.

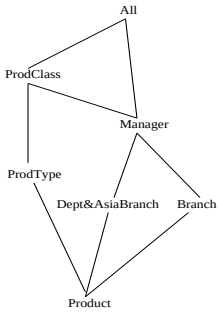
Different notions of schema equivalence have been around in the database field. The most general notion of schema equivalene, *absolute equivalence* [Hul86] characterizes the minimum requirements that two schemas must satisfy in order for them to have the same information capacity. Absolute equivalence is formalized by requiring the existence of a bijection between the instances of the schemas. Absolute equivalence is independent of the data model. A problem that arises with this notion of equivalence is that any arbitrary mapping may be used to guarantee absolute equivalence; furthermore, as observed by Miller et al. [MIR93], the mappings are not required to be finitely specifiable (they can be an infinite list of pairs of schema instances).

A hierarchy of more restricted notions of equivalence have been proposed [Hul86]. For example: *internal equivalence* requires the existence of a bijection that neither creates nor destroy elements in the instances; *query equivalence* requires the mappings to be expressible in the query language of the data model. We claim that they are of no practical use in the context of OLAP, because dimension instances with no intuitive relationship betweem tbetweenld be allowed to be associated via the mappings. This happens because these mapping do not necessarily preserve the hierarchical arrangement of OLAP data. We sustain that two dimension instances must be related via a mapping only if they have the same hierarchical domain. In order words, in the OLAP context, we conceive restructuring as a process in which we change the structure of the dimension (i.e. its hierarchy schema) without modifying its data hierarchy (hierarchical domain). As members are associated with facts in datacubes, this mapping restriction guarantees that the aggregate data are preserved, thus avoiding aggregate data re-computations, and keeping users to browse aggregate data using the same hierarchical domain.

As an example, Figure 2 depicts two possible hierarchy schemas to represent the instance on the right hand side of Figure 1. Both preserve the hierarchy domain. The constraints allow different graphs to represent the same information. For example, con-



(a) $\langle Product, Branch\rangle \oplus \langle Product, Department\rangle$
(b) $\langle Product, Branch\rangle \Leftrightarrow \langle Product, ProdType\rangle$
(c) $\langle Department, Manager, ProdClass\rangle$
(d) $\langle Branch = Asia\rangle \Leftrightarrow \langle Branch, Manager, ProdClass\rangle$



(a') $\langle Product, Dept\&AsiaBranch\rangle \oplus \langle Product, Branch\rangle$
(b') $(\langle Product, Branch\rangle \vee \langle Product, .., Dept\&AsiaBranch = Asia\rangle \Leftrightarrow \langle Product, ProdType\rangle$
(c') $\langle Dept\&AsiaBranch, Manager, ProdClass\rangle$

Figure 2: Product dimension schemas.

straints (c) and (d) for the hierarchy schema on the top translate to (c') for the hierarchy schema on the bottom. Observe that dimension on the top is homogeneous, i.e., every pair of members in the same category have ancestors in the same set of categories; the on in the bottom is heterogeneous because it models departments and branches in a single category called $Dept\&AsiaBranch$.

Contributions and Outline This paper proposes a notion of equivalence, hierarchical equivalence, which naturally captures dimension schema equivalence in OLAP. We prove that hierarchical equivalence can be characterized in terms of graph and schema isomorphisms in two known classes of dimension schemas, called here canonical and balanced. This result proves that canonical schemas are more expressive than balanced schemas, hence formally justifying the introduction of canonical schemas. We study hierarchical equivalence in dimension schemas enriched with dimension constraints. We present characterizations of hierarchical equivalence in terms of mapping between minimal dimensions contained by the schemas. We give a class of schemas –frozen schemas– that act as normal forms for dimensions schemas, in the sense that any dimension schema can be reduced via some well defined transformation to a unique (up to isomporhism) frozen schema. We prove that hierarchy equivalence test for frozen dimension reduces to a simple form of schema isomorphism, which leads to an algorithm for testing hierarchical equivalence. Additionally this result shows that schemas enriched with dimension constraints are more expressive than canonical schemas. Finally we prove complexity bounds and study algorithmic aspects of hierarchical equivalence testing.

The remainder of the paper is organized as follows. In Section 2 we review the main concepts related to schemas and state the notation. Section 3 introduces hierarchical equivalence and show its relation with balanced schemas. Section 4 studies hierarchical equivalence of canonical schemas, and shows that in this context hierarchical equivalence corresponds exactly with graph isomorphism of the corresponding hierarchy schemas. In Section 5 we generalize this result to dimension schemas, that is allowing to compare different hierarchy schemas and constraints. The notion of frozen schema is introduced and studied, along with the algorithmic aspects of hierarchical equivalence are studied. In Section 6 we present related work. Finally, in Section 7 we briefly conclude and outline further work. The complete proofs are presented in the appendix.

2 Preliminaries

2.1 Basic Graph Terminology

A (directed) graph G is a pair of sets (V, E) where $E \subseteq V \times V$. Elements $v \in V$ are called *vertices* and pairs $(u, v) \in E$ (directed) *edges*; u and v are *adjacent* vertices. A *path* in G from v to w is a sequence of vertices $v = v_0, \ldots, v_n = w$ such that $(v_i, v_{i+1}) \in E$. We say that v *reaches* w. The *length* of a path is n. A *cycle* is a path with $v = w$. A *dag* is a directed acyclic graph. A *sink* in a dag is a distinguished vertex w reachable from every other vertex in the graph. A *source* in a dag is a distinguished vertex v from which every other vertex of the graph is reachable. A *shortcut* in a dag is a path of length > 1 between two adjacent vertices. Given a vertex v of G, an *upgraph* is the subgraph of G generated by v and all the vertices reachable from it.

Given two graphs $G = (V, E)$ and $G' = (V', E')$, a *graph morphism* is a function $\phi : V \to V'$ preserving edges, that is, $(u, v) \in E$ implies $(\phi(u), \phi(v)) \in E'$. The morphism ϕ is called an *isomorphism (resp. monomorphism, epimorphism)* if ϕ as a function is bijective (resp. injective, onto).

2.2 Dimension Instance

Assume the existence of (possibly infinite) sets of categories $\mathbf{C}$, and of members $\mathbf{M}$.

Definition 1 (Hierarchy Schema) *A* hierarchy schema *is a* dag $H = (C, \nearrow)$, *where* $C \subseteq \mathbf{C}$, *having a distinguished category* $\texttt{All} \in C$ *which is a sink.*

Definition 2 (Hierarchy Domain) *A Hierarchy domain is a* dag $h = (M, <)$ *where* $M \subset \mathbf{M}$, *having a distinguished member* $\texttt{all} \in M$ *which is a sink, and without shortcuts.*

The last condition in Definition 2 (no shortcuts) avoids redundancies (transitive edges) in the representation of the data.

Given a child/parent relation $<$, we denote by $\ll$ the transitive closure of $<$. The reflexive and transitive closure of $<$, denoted $\leq$, is called *rollup relation*, and is a partial order between members.

Definition 3 (Dimension instance) *A* dimension instance d over a hierarchy schema $(C, \nearrow)$ *is a graph morphism* $d : (M, <) \to (C, \nearrow)$ *such that:*

1. $(M, <)$ *is a hierarchy domain;*

2. $d(\texttt{all}) = \texttt{All}$*;*

3. for all x *and* $y \neq z$*, if* $x \ll y \wedge x \ll z$ *then* $d(y) \neq d(z)$*.*

The fact that d is a graph morphism in Definition 3 states that whenever we have a child/parent relationship $m_1 < m_2$ between some pair of members $m_1 \in c_1$ and $m_2 \in c_2$, then there is an edge $c_1 \nearrow c_2$ in the hierarchy schema representing links between categories c_1 and c_2. Condition 3 of Definition 3 is a basic restriction in OLAP data modeling [HMV99, CT97, Kim97, LAW98], and states that the rollup relation $\leq$ is functional (i.e., single valued) between every pair of categories. This motivates to introduce the *rollup mapping* between two categories c_1 and c_2 of a dimension d, denoted $\Gamma_{c_1}^{c_2} d$, which is the restriction of $\leq$ to $d^{-1}(c_1)$ and $d^{-1}(c_2)$.

2.3 Dimension Schema

A dimension schema can be viewed as an abstract model of a dimension. It is used to visualize the data and to reason about summarizability. In previous work [HM02] we showed that the hierarchy schema itself is not enough expressive to support summarizability reasoning, and should be extended with constraints. This motivated us to introduce a class of constraints, *dimension constraints*, which together with the hierarchy schema forms a suitable schema to support summarizability reasoning. In this viewpoint, a dimension schema consists of a hierarchy schema, along with a set of *dimension constraints*. This notion of schema generalizes most well known classes of dimension schema. In what follows we formalize these notions.

Definition 4 (Dimension Constraint) *Let* $H = (C, \nearrow)$ *be a hierarchy schema,* $c \in C$*,* $K \subseteq \mathbf{M}$*. The language of constraints (with root* c*) has the following atoms:*

1. Path atoms: $\langle c, c_1, \cdots, c_n \rangle$*, where the* c_j *must satisfy that* $cc_1 \cdots c_n$ *is a path in* H*;*

2. Equality atoms: $\langle c, .., c' = k \rangle$*, where* c' *is such that there is a path from* c *to* c'*, and* $k \in K$*.*

A dimension constraint with root c *is a Boolean combination* ϕ *of atoms of the above kind.*

Dimension constraints consider the usual connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$, and $\oplus$ for exclusive disjunction. In addition, $\bot$ and $\top$ will denote the false and the true proposition, respectively.

Definition 5 (Semantics of Constraints) *Let* $d : (M, <) \to (C, \nearrow)$ *be a dimension instance, and* ϕ *a constraint with root* c*. Then* $d \models \phi$ *if and only if*

for all $m \in d^{-1}(c)$*,* $d \models \phi[c/m]$*,*

where $d \models \phi[c/m]$ *is defined recursively as follows:*

1. $d \models \langle c, c_1, \ldots, c_n \rangle [c/m]$ *iff there is a path* $mx_1 \cdots x_n$ *in* $(M, <)$ *with* $d(x_i) \in c_i$*.*

2. $d \models \langle c, .., c' = k \rangle [c/m]$ *iff* $d(k) \in c'$ *and* $m \leq k$.

3. $d \models (\phi \wedge \psi)[c/m]$ *iff* $d \models \phi[c/m]$ *and* $d \models \psi[c/m]$. *Similarly for* $\vee$ *and the other Boolean connectives.*

Given a hierarchy schema H and two sets of constraints Σ, Σ' over H, we say that Σ is equivalent to Σ', if for all dimension instances d over H it holds: $d \models \Sigma$ iff $d \models \Sigma'$.

Now we are ready to introduce the concept of Dimension Schema. The following definition extends Definition 3 in the presence of constraints.

Definition 6 (Dimension Schema) *A* dimension schema *is a pair* (H, Σ) *where* H *is a hierarchy schema and* Σ *is a set of constraints.*

A dimension instance d over a dimension schema $D = (H, \Sigma)$ *is a dimension instance* d *over* H *such that* $d \models \Sigma$. *The set of dimensions instances over* D *will be denoted by* $I(D)$.

Definition 7 (Schema Equiv. and Isom.) *Let* $D = (H, \Sigma)$ *and* $D' = (H', \Sigma')$ *be to dimension schemas.*

1. D *and* D' *are* equivalent, *denoted* $D \equiv D'$, *iff* $H = H'$ *and* Σ *is equivalent to* Σ'.

2. D *and* D' *are* isomorphic, *denoted* $D \cong D'$, *iff there exists a graph isomorphism* $f : H \to H'$ *such that* $(f(H), f(\Sigma)) \equiv (H', \Sigma')$, *where* $f(\Sigma)$ *stands for* Σ *modulo renaming by* f.

2.4 Classes of Dimension Schemas

The model we have presented subsumes the dimension models presented in the literature. The following definition formalizes two classes of dimension schemas that arise in OLAP.

Definition 8 (Classes of Dimension Schemas) *Let* $D = (H, \Sigma)$ *be a hierarchy schema.*

1. D *is* canonical *if* H *has no shortcuts and* Σ *is equivalent to* $\{\langle c, c' \rangle \mid c \nearrow c'\}$.

2. D *is* balanced *if* D *is canonical and* H *has a source.*

A dimension instance d is *homogeneous* if for every pair of categories $c_1 \nearrow c_2$ it holds that the rollup mapping $\Gamma_{c_1}^{c_2} d$ is a total function. Note that the constraint $\langle c, c' \rangle$ where $c \nearrow c'$ forces the rollup mapping from c to c' to be total. Therefore, canonical schemas convey all the homogeneous instances over its hierarchy schema. In this sense, in canonical schemas, Σ captures exactly homogeneity. Also notice that we have defined a canonical schema to be shortcut-free, because otherwise Σ would force the categories from which the shortcut start to be empty in every dimension conveyed by the schema. Balanced schemas correspond to the basic class of schemas introduced in early works on OLAP. They are the logical representation of dimension schemas in early snowflake schemas [CD97]. Canonical schemas were introduced by Jagadish et al. [JLS99] to overcome some of the weaknesses of balanced schemas. Canonical schemas allow unbalancedness, that is, they can have dimension instances with two members in the bottom categories having ancestors in different sets of categories. This has been shown to be an important feature to provide flexibility in OLAP modeling.

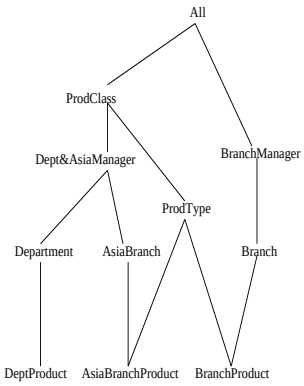


Figure 3: A canonical schema for the bank products.

Example 1 *Figure 2.4 depicts the hierrachy schema of canonical schema for modeling the bank products. The set of dimension constraints has a constraint* $\langle c, c' \rangle$, *for all edges* (c, c') *in the hierarchy schema. Observe that the products are now split in three categories depending on where their roll up (only to Department, to AsiaBranch and ProdType, etc.), allowing to think of as if there were one hierarchy (the upgraph) for each kind of product. This illustrates the flexibility in modeling given by canonical schemas.*

Given two classes of schemas S_1, S_2, we define $S_1 \sqsubset S_2$ iff for each schema in S_1, there is an equivalent schema in S_2. Then it holds Balanced Schemas $\sqsubset$ Canonical Schemas $\sqsubset$ Dimension Schemas.

3 Hierarchical Equivalence

In this section we present the notion of hierarchical equivalence in which dimension schemas are related via mappings that preserve the hierarchical domain of the dimensions.

The following definition generalizes Definition 7 for schemas over arbitrary hierarchy schemas.

Definition 9 (Hierarchical Equivalence) *Two dimension shemas D and D' are* hierarchically equivalent *(h-equivalent) if and only if there is a bijective function $f : I(D) \to I(D')$ such that for all $d \in I(D)$, $dom(d) = dom(f(d))$. In this case we write $D \equiv_h D'$.*

Observe that the relation $\equiv_h$ is an equivalence relation. Also, it is worth noting that the instance mapping f required for h-equivalence is *internal* [Hul86], i.e., it does neither create nor destroy members or constants in the instances. Moreover, the mapping is *generic* [Hul86], that is, given a pair of dimension instances d and d' with $d' = f(d)$, if we apply the same permutation π of members to d and to d', if $\pi(d)$ is in the domain of f then $\pi(d') = f(\pi(d))$.

Example 2 *Consider the dimension schemas $D_1 = (A, \Sigma_1)$, $D_2 = (B, \Sigma_2)$ and $D_3 = (C, \Sigma_3)$, where A, B, C are the hierarchy schemas in Figure 4, $\Sigma_1 = \Sigma_3 = \emptyset$ and $\Sigma_2 = \{\neg\langle e, f\rangle \vee \neg\langle e, g\rangle\}$. Then $D_1 \equiv_h D_2$ via mapping the members of c to f, the members of d to g, and the members of a and b to e. However, it is not the case that $D_1 \equiv_h D_3$. Indeed, given a member m, there is a unique dimension instance in $I(D_3)$ whose child/parent relation is $\{m < \texttt{all}\}$, but there are two dimension instances in $\mathcal{I}(D_2)$ whose child/parent relation is $\{m < \texttt{all}\}$.*

It is not difficult to check that if $D \equiv D'$ then $D \equiv_h D'$. We end this section by showing that it is straightforward to show that the converse also holds for balanced schemas.

A dimension instance d is *exact* if d is bijective. It is easily verified that all canonical dimension schemas have an exact dimension instance.

Theorem 1 (h-Equiv. of Balanced Schemas) *Two balanced dimension schemas $D = (H, \Sigma)$ and $D' = (H', \Sigma')$ are h-equivalent if and only if H and H' are (graph) isomorphic.*

Proof of Theorem 1 One direction is obvious.

Assume that $D \equiv_h D'$ via f. Consider an exact dimension d of D. Then, as graphs, $H \cong \text{dom}(d) \cong \text{dom}(f(d))$. Now, because D' is balanced there is a (graph) monomorphism $\mu : \text{dom}(f(d)) \to H'$ with $\mu(\texttt{all}) = \texttt{All}$ (if $\mu(v) = \mu(w)$ for $v \neq w$, the source of $\text{dom}(f(d))$ would have two ancestors in the same category, violating condition 3 of Definition 3.) Hence there is a monomorphism $H \to H'$. By the same argument on the reverse direction, there is a monomorphism $H' \to H$. Hence $H \cong H'$. □

4 Hierarchical Equivalence of Canonical Schemas

This sections extends the results of Theorem 1 to canonical dimensions. The importance of this result is twofold: (1) The notion of h-equivalence has a simple and intuitive characterization as graph isomorphism. (2) This proves that canonical schemas are strictly more expressive than balanced schemas (because given a canonical and not balanced schema there is no balanced schema isomorphic to it.) So we have now a formal argument that justifies the introduction of canonical schemas for OLAP modeling.

First, observe that the argument in the proof of Proposition 1 does not necessarily work in this setting (there could be no injective μ).

4.1 Hierarchical Equivalence and Isomorphism

The following is the main result of the section.

Theorem 2 (h-Equiv. of Canonical Schemas) *Let $D = (H, \Sigma)$ and $D' = (H', \Sigma')$ be two canonical schemas. Then, $D \equiv_h D'$ if and only if H is (graph) isomorphic to H'.*

Proof of Theorem 2 Let us sketch the non-trivial direction of the proof. Let $H = (C, \nearrow)$ and $H' = (C', \nearrow')$ and $f : I(D) \to I(D')$ be the bijection given by $\equiv_h$.

(*) Let $d_1 : (M, <) \to H$ be an exact dimension of D (hence $H \cong (M, <)$). Let $f(d_1) : (M, <) \to H'$ be the image of d_1 under f (by hypothesis $f(d_1)$ has the same domain of d_1). Let d'_1 be an exact dimension of $f(d_1)(M)$. Let d_2 be an exact dimension of $f^{-1}(d'_1)$. Continue this process until $H_1 = \text{Im}(d_i) \cong \text{Im}(d'_i) = H'_1$. Denote by μ_1 this isomorphism. Note that H_1 is well defined because the process terminates by a graph theoretic argument.

For each dimension instance $d : M_1 \to H$ with $d(M_1) = H_1$ do: Redefine f by performing the following operations: $y := f(d)$; $f(d) := (\mu_1 \circ d)$; $ff^{-1}(\mu_1 \circ d) := y$. Recall from Section 2 that an instance d takes its domain from a possibly infinite set $\mathbf{M}$. Here we assume that the set $\mathbf{M}$ is finite, hence the loop ends. The extension to the infinite case is straightforward. It is easily verified that at the end of this process we will have that for all complete d of H_1, it holds that $f(d) = (\mu_1 \circ d)$. Call f_1 this new f.

Now we repeat the whole process starting from (*) with f_1. This process generates a H_2, H'_2 and a new f_2.

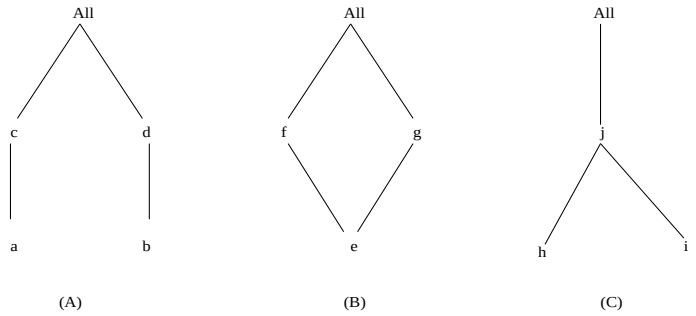


Figure 4: Three hierarchy schemas.

Observe that $H_2 \neq H_1$, because otherwise there must be a complete dimension d of H_1 which is not map to the complete dimension $(\mu_1 \circ d)$ via f_1.

By repeating this process we generate a series $(H_1, H_1', f_1), (H_2, H_2', f_2), \ldots$ This series has the property $H_i \neq H_j$ for $i \neq j$ by an argument similar to the case $i = 1$.

Finally just note that this series must be infinite. □

The following examples illustrates the main idea of the previous proof.

Example 3 *Let D and D' be the dimension schemas of Figures 5(A) and 5(B). Clearly they are not graph isomorphic. Assume that $D \equiv_h D'$ via an instance mapping f. Figure 6 depicts, on the top, the triple (H_1, H_1', f_1), and in the bottom (H_2, H_2', f_2), in a possible sequence generated in the proof for D and D'. f_1 sends the complete dimension of the subschema underlined to the one underlined in H_1'. Similarly for f_2. This property forces the schema H_2 (resp. H_2') to be different from H_1 (resp. H_1'). This series is infinite, but it can be checked now that there is no next triple (H_3, H_3', f_3), yielding a contradiction. Hence $D \not\equiv_h D'$.*

5 Hierarchical Equivalence in Dimension Schemas

In this section we present a characterization of h-equivalence for dimension schemas, which yields an algorithm for testing h-equivalence.

5.1 Frozen Equivalence

We introduce a notion of equivalence, *frozen equivalence*, defined in terms of injective mappings between a special kind of dimension instances, called frozen. Intuitively, a *frozen dimension* is a minimal model conveyed by a dimension schema. (We refer the reader to previous work [HM02] for details.) The notion of frozen dimension is essential for giving an algorithmic version of h-equivalence.

Let $D = (H, \Sigma)$ and $H = (C, \nearrow)$. We need to define two functions, $\texttt{NotKnown} : \mathbf{C} \to \mathbf{M}$, an injective function that assigns a fix member to each category, and $\texttt{Const}_D : C \to \mathbf{K}$, defined by $\texttt{Const}_D(c)$ to be the set of constants k appearing in Σ.

Definition 10 (Frozen Dimension) *Given a dimension schema D and $c \in C$, a* frozen dimension *with root c is a dimension instance $d : (M, <) \to (C, <)$ such that:*

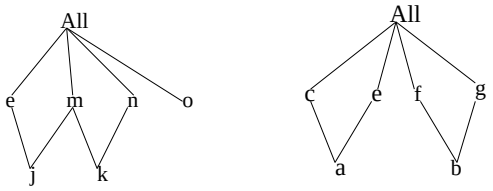
1. *d is injective;*
2. *$c \in Im(d)$;*
3. *$d^{-1}(c)$ is a source of $(M, <)$;*
4. *For all $x \in M$, $x \in \texttt{Const}_D(d(x)) \cup \texttt{NotKnown}(d(x))$.*

We denote by $\texttt{Frozen}(D, c)$ the set of frozen dimension of D with root c, and by $\texttt{Frozen}(D)$ the union of all $\texttt{Frozen}(D, c)$ for all categories c of D.

Example 4 *Figure 7 depicts the frozen relation between the product schemas of Figure 2. (We are only showing the frozen dimensions with root Product.)*

Definition 11 (Frozen Equivalence) *Given a pair of dimension schemas $D = (H, \Sigma)$ and $D' = (H', \Sigma')$, where $H = (C, \nearrow)$ and $H' = (C', \nearrow')$:*

1. *A* category correspondence *between D and D' is a binary relation $\Omega \subseteq C \times C'$.*
2. *A correspondence $\Omega \subseteq D \times D'$ induces a binary relation $\mathcal{F}_\Omega \subseteq \texttt{Frozen}(D) \times \texttt{Frozen}(D')$, called* frozen relation, *defined as the pairs $(d, d') \in \texttt{Frozen}(D) \times \texttt{Frozen}(D')$ such that there is a graph isomorphism $\mu : Im(d) \to Im(d')$ with $\mu \subseteq \Omega$, and for each $c \in C$ and $k \in \texttt{Const}_D(c)$, $d(k) = c$ implies $d'(k) = \mu(c)$. In this case we say that $d \cong d'$ (frozen dimension isomorphism).*
3. *Two schemas D and D' are frozen equivalent (in the sequel f-equivalent) if there is a bijective frozen relation between them.*



(A) (B)

Figure 5: Two hierarchy schemas.

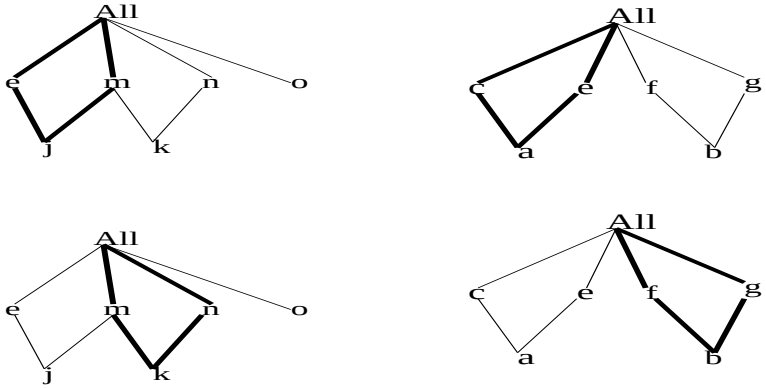


Figure 6: A series of matchings illustrating proof of Theorem 2

Proposition 1 (f-Equiv. implies h-Equiv.) *Let D and D' be two dimension schemas. If D and D' are f-equivalent, then D and D' are h-equivalent.*

Proof of Proposition 1 The proof of this Proposition builds a bijective mapping $f : I(D) \rightarrow I(D')$ using the bijective frozen relation $r : \texttt{Frozen}(D) \rightarrow \texttt{Frozen}(D')$.

Now we define Function f. Given a dimension instance $d : (M, <) \rightarrow (C, \nearrow)$, such that $d \in I(D)$, $d' : (M, <) = f(d)$ is defined as follows: for every member x in M:

(1) let e be d restricted to the upgraph of $(M, <)$ with source x; It holds that $e \in I(D)$ (e is a *dimension tuple* and it has been proved [Hur02] that if $d \in I(D)$, all the dimension tuples of d are also in $I(D)$)

(2) for each member y in $\mathrm{dom}(e)$ we rename y with $\texttt{NotKnown}(e(y))$ if $y \notin \texttt{Const}_D(e(y))$, obtaining the dimension instance g_x. It is easily verified by inspection that g_x is a frozen dimension of D with root $c = d(x)$;

(3) We have that $g_x \cong r(g_x)$ via some isomorphism $\mu_x \subseteq \Omega$, where Ω is the category correspondence that defines r. Then, we have $d'(x) = \mu_x(c)$.

Notices that steps 1-3 above define $c' = d'(x)$, for each member x in M.

The above procedure has the following properties: For each pair of members x_1 and x_2 in M: (P1) if $x_1 \leq x_2$ then $\mu_{x_1} \subseteq \mu_{x_2}$; and (P2) if $x_1 < x_2$ then $\mu_{x_1}(d(x_1)) \nearrow' \mu_{x_2}(d(x_2))$. Property 1 follows from the fact that if $x_1 \leq x_2$, $g_{x_1} \subseteq g_{x_2}$. Property 2 follows from the fact that if $x_1 < x_2$, the source of $\mathrm{dom}(g_{x_1})$ is a child, in the dimension g_{x_2}, of the source of $\mathrm{dom}(g_{x_2})$.

It is easily verified that d' is unique. Now, we prove that d' is a dimension instance over H, i.e., we prove that d' is a graph morphism and it satisfies conditions 1-3 of Definition 3. Assume d' is not a graph morphism, then there are members $x_1 < x_2 \in M$ such that $d'(x_1) \nearrow' d'(x_2)$ does not hold. Thus $\mu_{x_1}(d(x_1)) \nearrow' \mu_{x_2}(d_{x_2})$ does not hold, contradicting Property 2. Conditions 1-2 of Definition 3 are easy to verify. Now, assume d' does not satisfy Condition 3, then there are members x, y, z in M, such that $x < y$, $x < z$ and $d'(y) = d'(z)$. Then, $\mu_y(d(y)) = \mu_z(d(z))$. Then, by Property 1, we have that $\mu_x(d(y)) = \mu_x(d(z))$. But, because d satisfies Condition 3 of Definition 3, $d(y) \neq d(z)$, and hence μ_x is not bijective (and not an isomorphism), yielding a contradiction.

Now, we prove that $d' \models \Sigma$, assume not, then by a basic property of dimension schemas proved in previous work [HM02], there must be some dimension tuple e of d', such that $e \not\models \Sigma$. Now, we obtain a dimension g from e similarly as done in Step 2 above. Because $e \not\models \Sigma$, we have that $g \not\models \Sigma$. By construction

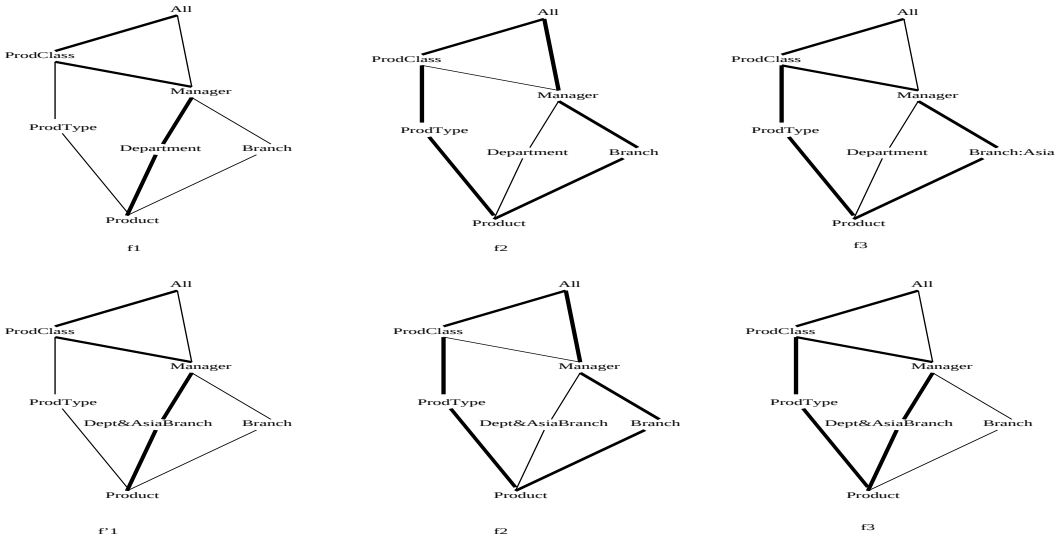


Figure 7: Frozen relation

of d', g must be in $\mathrm{Im}(r)$, thus $g \in \mathtt{Frozen}(D')$, and $g \models \Sigma$, leading to a contradiction.

So far, we have proved that f is well defined, i.e., f is a function (thus total) from $I(D)$ to $I(D')$. It remains to prove that f is surjective and injective. That is, f^{-1} is also a function. This follows from the fact that, applying steps 1-2, modulo replacing D with D', and vice versa, we obtain f^{-1} (instead of f). □

If we pre-compute and keep a frozen mapping stored, the computation of $f(d)$ (see proof of Proposition 1) can be accomplished in polytime. Consequently, we may compute the instance mapping induced by a bijective frozen relation in polytime in the size of the input dimension instance and the frozen relation.

In Section 5.3 we will state and prove the converse of Proposition 1.

5.2 Frozen Schemas

Firstly, we will introduce *frozen schemas*, dimension schemas that are normal forms, in the sense that every dimension schema is h-equivalent to a frozen schema.

Definition 12 (Frozen Schema) *A* frozen schema *is a dimension schema D such that each category c in D has a unique frozen dimension d and $Im(d)$ is exactly the upgraph of c.*

Notice that a frozen schema does not have shortcuts. Also, it is easily verified that canonical schemas are frozen schemas.

Example 5 *The bottom schema given in Figure 2 is a frozen schema. The category AsiaBranch is the only category whose frozen dimension has a constant (Asia) different than* $\mathtt{NotKnown}(AsiaBranch)$.

Next, we show that testing h-equivalence of frozen schemas defined over the same set of constants reduces to testing whether the schemas are isomorphic. This result generalizes Theorem 2 because canonical schemas are frozen schemas.

Theorem 3 (h-Equiv. in Frozen Schemas) *Let D and D' be two frozen dimension schemas such that for all $c \in C$ and $c' \in C'$ it holds $\mathtt{Const}_D(c) = \mathtt{Const}_{D'}(c')$. Then D and D' are h-equivalent iff they are isomorphic (i.e., $D \equiv_h D'$ iff $D \cong D'$).*

Proof of Theorem 3 The proof is a generalization of the proof of Theorem 2. □

This theorem also shows that dimension schemas are more expressive than canonical schemas because some frozen schemas are not isomorphic to any canonical schema.

5.3 h-Equivalence and f-Equivalence

In this section we show that h-equivalence implies f-equivalence. This result along with Proposition 1 shows that f-equivalence characterizes h-equivalence.

Theorem 4 (h-Equiv. in dimension Schemas) *Let D and D' be two frozen dimension schemas such that for all $c \in C$ and $c' \in C'$ it holds $\mathtt{Const}_D(c) = \mathtt{Const}_{D'}(c')$. Then D and D' are f-equivalent iff they are h-equivalent.*

Proof of Theorem 4 One direction is Proposition 1.

So assume that D and D' are h-equivalent. First define a schema transformation that takes D and produces a frozen schema D_f h-equivalent to D. The transformation works as follows: (1) Compute the frozen dimensions of D using the DIMSAT algorithm in previous work [HM02]; (2) Do a topological sort of the graph $(C, \nearrow')$ with the edges reversed. (3) Following the topological sort, for each category c with more than one frozen dimension, split c into $c_1, \ldots, c_n$ (preserving adjacent edges) yielding a new hierarchy schema with a single frozen dimension in each c_j; (4) For each c_j delete adjacent edges that do not match the frozen dimension.

Each split in step (3) induces induces the following category correspondence between the hierarchy schemas before and after the split: (c, c_j) for all $1 \leq j \leq n$, and for the remaining categories c' that appear in both hierarchy schemas we have (c, c').

It is not difficult to verify that this category correspondence induces a bijective frozen mapping between the old and the new schema. By composing these frozen mappings we get a bijective frozen mapping between D and D_f.

In the same manner, we build a frozen schema D'_f and a bijective frozen mapping between D' and D'_f. Hence, we have bijective frozen mappings $D \to D_f$ and $D' \to D'_f$. Also, from Theorem 3 we know that $D_f \cong D'_f$ via some μ. From μ we can build a bijective frozen relation between D_f and D'_f. Composing these mappings we get the statement of the theorem. □

Example 6 *The bijective frozen relation of Figure 7 shows that the two schemas of Figure 2 are h-equivalent.*

5.4 Algorithmic aspects

From the proof of Theorem 4, we can derive an algorithm for testing h-equivalence and prove that this problem is decidable. The naive application of the procedure in the proof yields a double exponential time algorithm. In fact, we can test whether $D \equiv_h D'$ in the following two steps:

1. Apply the transformation in step (4) in the proof to transform D into D_f and D' into D'_f;

2. Test whether H_f is graph isomorphic to H'_f, where H_f (resp. H'_f) is the hierarchy schema of D_f (resp. D'_f).

The number of categories in the frozen schemas is in $O(n2^nK)$, where n is the number of categories and K is the number of constants mentioned in the schema. This bound is the order of the number of splits used in each transformation. Essentially, we may have as many categories in the resulting schema as frozen dimensions in the original schema. Since the size of D_f (resp. D'_f) is exponential in the size of the initial schema D (resp. D') we get the stated bound due to the test in 2.[1]

The following result shows that the problem is hard.

Theorem 5 (Lower bound for Testing h-Equiv.) *Testing whether two dimension schemas D and D' are h-equivalent is co-NP hard.*

Proof of Theorem 5 We will present a polytime transformation from VALIDITY, which is known to be CoNP-complete. In VALIDITY we are given a proposition P and we are asked whether P is valid, i.e., whether P is satisfied by all truth assignments. From an instance P of VALIDITY we obtain the dimension schemas D and D'. Both schemas have the same hierarchy schema H with a bottom category c_b, a top category `All`, and a set C_p containing one category c_p for each propositional variable p in P. In addition, c_b is connected to every category in C_p, and every category in C_p is connected to `All`. The schema D has a unique constraint $\bot$ (false), and D' has a single constraint $\neg\alpha_P$, where α_P is the dimension constraint obtained from P by replacing each propositional variable p with $\langle c_b, c_p \rangle$. Now, we show that $D \equiv_h D'$ iff P is valid. (If) If P is valid, c_b is unsatisfiable in D', then the schemas are equivalent and thus h-equivalent. (Only If) If the schemas are h-equivalent and P is not valid, c_b is satisfiable in D'. Thus, the instances d with members in $d^{-1}(c_b)$ cannot be mapped to instances of D, yielding a contradiction. □

We end this section by sketching an exponential time algorithm for testing h-equivalence.

1. Compute the frozen dimensions of D and D';

2. For every binary relation between categories, test whether it induces a bijective frozen mapping.

Step 1 can be done in exponential time in the size of the schema. (See [HM02] for detailed bounds.) The number of binary relations between categories we need to test in Step 2 is $O(2^{n^2})$. For each such relation, we have to compute the induced frozen relation R, i.e. we need to test for each pair of frozen dimensions $d \in \texttt{Frozen}(D)$ and $d' \in \texttt{Frozen}(D')$ whether $(d, d') \in R$. This test can be done in 2^{n^2} operations of $O(n)$ steps each, since we need to check at most

[1] DAG isomorphism is graph isomorphism complete. Recall that the "exact" complexity of deciding whether two graphs are isomorphic is still not known. The problem has neither been proved to be NP complete nor in P.

2^{n^2} possible isomorphisms between d and d'. Also, we have to perform one test for each pair of frozen dimensions d and d'. Since the number of frozen dimensions of a given schema is exponential in the size of the schema, Step 2 can be accomplished in time exponential in the size of the schemas.

In Section 5.3, we showed that a bijective frozen relation induces a bijection between the instances of the schemas, and that we may implement this function as a polytime function in the size of the instance and frozen relation. Thus, it is always possible to efficiently translate instances between two h-equivalent schemas.

6 Related Work

There has been abundant work on OLAP dimension modeling over the past few years [CT97, HMV99, LAW98, PJE99, JLS99]. However, to the best of our knowledge, there are no studies regarding dimension schema equivalence. Other notions of equivalence and their testing have been studied for generic graph data models by Miller et al. [MIR94] and nested data models [VL00]. Several sets of schema transformations [MIR94, RR98] have proven successful in supporting the restructuring of schemas in a variety of data models. These notions are not suitable for restructuring data in OLAP for the same reasons given before.

Furthermore, Hurtado [Hur02] shows that dimension constraints although being first order constraints, are orthogonal to traditional constraints studied in the database literature [AV97] (the extra expressiveness is needed to support summarizability reasoning). As the test for equivalence depends on the class of constraints the models have, the problem we address in this paper is not related to previous work on schema equivalence. In the next two paragraphs we explain this point into more detail.

Let us first explain the relationship between dimension constraints and First Order Logic (FOL) constraints, that may expressed over the relational representation described above. An important property of the hierarchical domain $<$ of a dimension instance is that the size of its largest path should be smaller than the size of the largest path without cycle in the hierarchy schema. This turns the ancestor/descendant relation $\ll$ to be FOL definable. Consequently, the conditions that a child/parent relation must satisfy and the conditions 3 in Definition 3 are FOL definable. In addition, it is easily verified that dimension constraints are FOL constraints; therefore, dimension constraints along with the partitioning property may be expressed as FOL constraints over a snowflake representation of the dimension instance.

Abiteboul et al. [AV97] study a class of FOL constraints called *embedded constraints* that formalizes a wide variety of constraints studied in the database literature. Embedded constraints essentially say that the presence of some tuples in the instance implies the presence of some tuples in the instance or implies that certain tuple components are equal. Dimension constraints cannot be expressed with embedded constraints, since we cannot express with them constraints that assert dependences such as "some tuples or some other tuples appear in the instance".

Example 7 *Consider the dimension constraint* $\langle c, c_1\rangle \vee \langle c, c_2\rangle$. *This constraint is equivalent to the following FOL expression:*

$$\forall x(d^{-1}(c)(x) \Rightarrow \exists x_1 \exists x_2(\Gamma_c^{c_1}(x, x_1) \vee \Gamma_c^{c_2}(x, x_2))).$$

This constraint cannot be expressed with an embedded constraint, since an embedded constraint is an expression of the form

$$\forall x_1, \ldots, x_n(\phi(x_1, \ldots, x_n) \Rightarrow \exists z_1, \ldots, z_k \psi(y_1, \ldots, y_m)),$$

where $\{z_1, \ldots, z_k\} = \{y_1, \ldots, y_m\} \perp \{x_1, \ldots, x_n\}$, *and* ϕ *and* ψ *are conjunctions of atoms.*

Some researchers have considered the problem of restructuring multidimensional OLAP data. Gyssens and Lakshmanan [GL96] proposed restructuring operators that interchange categories and measures in fact tables without losing information content. Using these operators, it is possible to drop, add, or rename a category; drop or add measures; and to change a category to a measure and vice versa. Similar operators are introduced by Gupta et al. [GHQ95]. In these approaches, dimension hierarchies are not modeled explicitly and their results are hence orthogonal to the problem of restructuring dimensions. Lehner et al. [LAW98] model a class of dimension schemas that allows structural heterogeneity, and propose an operator for transforming them into balanced schemas, which they say to be in *dimensional normal form* (DNF). The transformation is done by treating categories causing heterogeneity as attributes, which flattens the hierarchical structure of the schema, causing a loss of information capacity.

All these works are over setting where no formal notion of schema equivalence is provided.

Pedersen et al. [PJE99] propose instance mappings to transform heterogeneous dimensions into homogeneous dimensions by adding null members (thus the mappings are non-internal mappings). These mappings are used to normalize dimensions in order to

apply traditional OLAP summarizability reasoning. The proposed mappings, however, are only applicable to a restricted class of dimension instances. An extension of the mappings to deal with the general case of heterogeneity may generate very intricate dimension instances, with lots of null values, and many-to-many rollup mappings (thus violating Condition 3 of Definition 3). (For further details see [Hur02].) In this paper, we have proved that any dimension schema may be transformed into a frozen schema (which conveys only homogeneous dimensions), and that their instances can be related via a polytime computable mapping which does not add null values and preserves hierarchical domains.

7 Conclusion and Further Work

In this paper we have presented a series of results that give conceptual insights into the problem of modeling OLAP hierarchies. In particular, our framework: allowed us to compare different classes of dimension schemas introduced in a variety of OLAP models; and provides a formal basis to further research on schema integration and restructuring in OLAP warehouses.

Further work includes the definition of normal forms, restructuring operators, notion of information dominance and implementation issues.

References

[Alb00] J. Albert. Theoretical foundations of schema restructuring in heterogeneous multidatabase systems. In *Proceedings of the ACM Conference on Information and Knowledge Management*, Washington, DC, USA, 2000.

[AV97] S. Abiteboul and V. Vianu. Regular path queries with path constraints. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, Tucson, Arizona, USA, 1997.

[CD97] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. In *ACM SIGMOD Record 26(1)*, March 1997.

[CT97] L. Cabibbo and R. Torlone. Querying multidimensional databases. In *Proceedings of the 6th International Workshop on Database Programming Languages*, East Park, Colorado, USA, 1997.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass. Aggregate query processing in data warehousing environments. In *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, 1995.

[GL96] M. Gyssens and L. Lakshmanan. A foundation for multi-dimensional databases. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, Bombay, India, 1996.

[HM01] C. Hurtado and A. Mendelzon. Reasoning about summarizability in heterogeneous multidimensional schemas. In *Proceedings of the 8th International Conference on Database Theory*, London, UK, 2001.

[HM02] C. Hurtado and A. Mendelzon. OLAP dimension constraints. In *Proc. PODS 2002*, Madison, USA, 2002.

[HMV99] C. Hurtado, A. Mendelzon, and A. Vaisman. Maintaining data cubes under dimension updates. In *Proceedings of the 15th IEEE International Conference on Data Engineering (ICDE)*, Sydney, Australia, 1999.

[Hul86] R. Hull. Relative information capacity of simple relational database schemata. In *SIAM Journal of Computing 15(3):865-886*, 1986.

[Hur02] C. Hurtado. Structurally heterogeneous OLAP dimensions. In *Doctoral Thesis, Computer Science Dep. Toronto*, 2002.

[JLS99] H. V. Jagadish, L. V. S. Lakshmanan, and D. Srivastava. What can hierarchies do for data warehousesΓ In *Proc. of the 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland, UK, 1999.

[Kim97] R. Kimball. A dimensional modeling manifesto. *DBMS and Internet Systems Magazine, http://www.dbmsmag.com*, August 1997.

[LAW98] W. Lehner, H. Albrecht, and H. Wedekind. Multidimensional normal forms. In *Proceedings of the 10th Statistical and Sci-*

entific Database Management Conference, Capri, Italy., 1998.

[MIR93] R. Miller, Y. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *Proceedings of the 19th International Conference on Very Large Data Bases (VLDB)*, Dublin, Ireland, 1993.

[MIR94] R. Miller, Y. Ioannidis, and R. Ramakrishnan. Schema equivalence in heterogeneous systems: Bridging theory and practice. In *Information Systems, vol. 19, no.1*, 1994.

[PJE99] T. B. Pedersen, C. S. Jensen, and Dyreson C. E. Extending practical preaggregation in on-line analytical processing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland, 1999.

[RR98] S. Ram and V. Ramanesh. Schema integration: Past, present, and future. In *A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, editors, Management of Heterogeneous and AutonomousDatabase Systems. Morgan-Kaufmann*, San Mateo,C.A., USA, 1998.

[VL00] Millist W. Vincent. and Mark Levene. Restructuring partitioned normal form relations without information loss. *SIAM Journal on Computing*, 29(5):1550–1567, 2000.