

# Interfaces to Data, Beyond Views

Michael Benedikt<sup>1</sup>

<sup>1</sup>Oxford University, United Kingdom

## Abstract

We discuss different ways in which we can define interfaces to data, going beyond standard view definitions. One is based on specifying a query, and asking for the views within a class that give the minimal information that suffice to support the query. Another is based on specifying which instances should be indistinguishable to users via the interface.

## 1. Introduction

How can we define an interface to a relational structure? In databases, a standard means is via *views* – declarative queries whose output is made available to users as another structure: in the case of relational database queries, a table. A huge amount of study has gone into the properties of views and view languages. See e.g. [1].

There are other mechanisms for interfacing with data that differ from views. At certain times in the past, defining more flexible notions of “datasource capabilities” – one aspect of how queries can interface with a data source – represented quite a hot topic in data management [2]. One kind of data interface that has received continued attention are *access methods* [3, 4, 5, 6, 7]. Access methods give a functional interface allowing a user to look up matching rows in a table, providing as input a subset of the attributes. They restrict access in a manner orthogonal to views, but one can add to their expressiveness by combining them with integrity constraints [8] or with views [9]. Or one can limit the expressiveness further by adding number restrictions on access outputs [10]. Cautis et. al [11] give an intriguing formalism for defining “infinite sets of views”, an interface limitation that is incomparable to access restrictions.

Since this kind of research has diminished over the last few years, in this article we want to overview a couple projects that might stimulate new thinking about interfaces to data beyond views, and how one might measure the expressiveness of interfaces. This will be based on work published previously [12], and also connected to work that is about to appear [13].

## 2. Distributed schemas and classical views

We first review some standard definitions in relational databases [14]. A *schema* consists of a finite set of relation names, each with an associated arity (a non-negative number). An *instance* of a schema is an assignment of each relation name  $R$  in the schema of arity  $n$  to a collection

---

AMW 2024: 16th Alberto Mendelzon International Workshop on Foundations of Data Management, September 30th–October 4th, 2024, Mexico City, Mexico

✉ michael.benedikt@cs.ox.ac.uk (M. Benedikt)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

(finite or infinite) of  $n$ -tuples of elements. We will often use  $\mathcal{I}$  for a generic instance. The *active domain* of an instance  $\mathcal{I}$ , denoted  $\text{adom}(\mathcal{I})$ , is the set of elements occurring in some tuple of some  $\mathcal{I}(R)$ .

An  $n$ -ary *query* is a function from instances of a fixed schema  $\mathcal{S}$  to  $n$ -tuples. We assume the usual semantics for active-domain first-order logic, given by a relation  $\mathcal{I} \models \gamma$  for  $\gamma$  a sentence. A *Conjunctive Query* (CQ) is a logical formula of the form  $\exists \mathbf{y}. \bigwedge_i A_i$  where  $A_i$  are atoms over the schema. A *Union of CQs* (UCQ) is a disjunction of CQs where the free variables of each disjunct are the same. UCQs can be extended to *relational algebra*, the standard algebraic presentation of first-order relational queries: queries are build up from symbols for each relation name by union, difference, selection, and product [14]. Relational algebra queries can be expressed in first-order logic. Unlike arbitrary first-order logic formulas, whose truth value may depend on an additional “universe” for the range of quantification, relational algebra queries are *domain independent*: the output on an instance depends on the interpretation of each relation symbol. A *view* is just a pairing of an  $n$ -ary query with an  $n$ -ary relation that names its output. Based on the kind of query, we can talk about a *CQ view*, a *relational algebra view*, etc.

We now come to some definitions that are specific to distributed query processing or data integration. A *distributed schema* (d-schema)  $\mathcal{S}$  consists of a finite set of *sources*  $\text{Srcs}$ , with each source  $s$  associated with a *local schema*  $\mathcal{S}^s$ .

A query over a d-schema is simply a query over the relations occurring in some local schema. A *distributed view* (d-view) assigns to each local schema a set of views over that schema. Thus a d-view describes an interface where each local source exports some information.

**Example 1.** We have a d-schema consisting of two sources, a data source representing papers submitted to VLDB VLDB(paperid, title, authorid, authorname) and another source with relation SIGMOD with the same attributes, containing papers submitted to SIGMOD.

A trivial *d-view* might export the entire content of each relation. Another *d-view* might have the VLDB source export the projection on paperid, and the SIGMOD source export authorids. Note that a query giving the intersection of VLDB and SIGMOD is *not* a d-view, since a d-view consists of views that are local to a source.  $\triangleleft$

### 3. Minimal information interfaces

We are ready to look at our first non-standard means of defining an interface, from [12]. Rather than having the data designer explicitly provide views, the idea is that they only specify a query, and then ask for the *minimal information views* (within a certain class) that can support answering that query. This is particularly interesting in the distributed setting. We want to formalize the notion of a set of views being *useful* for answering a query, and having *minimal information* among those that are useful. We use Segoufin and Vianu’s notion of *determinacy*, later developed jointly with Nash [15].

**Definition 1.** Two *d-instances*  $\mathcal{D}$  and  $\mathcal{D}'$  are indistinguishable by a *d-view*  $\mathcal{V}$  (or just  $\mathcal{V}$ -indistinguishable) if  $V(\mathcal{D}) = V(\mathcal{D}')$  for each view  $V \in \mathcal{V}$ .

Since each view  $V \in \mathcal{V}^s$  is defined only using relation names occurring in  $\mathcal{V}^s$ , we can equivalently say that  $\mathcal{D}_1$  and  $\mathcal{D}_2$  are  $\mathcal{V}$ -indistinguishable if  $V(\mathcal{D}_1^s) = V(\mathcal{D}_2^s)$  for each  $V \in \mathcal{V}^s$  for each source  $s$ .

**Definition 2.** A  $d$ -view  $\mathcal{V}$  determines a query  $Q$  at a  $d$ -instance  $\mathcal{D}$  if  $Q(\mathcal{D}') = Q(\mathcal{D})$  for each  $\mathcal{D}'$  that is  $\mathcal{V}$ -indistinguishable from  $\mathcal{D}$ .

The  $d$ -view  $\mathcal{V}$  is useful for  $Q$  if  $\mathcal{V}$  determines  $Q$  on every  $d$ -instance (for short, just “ $\mathcal{V}$  determines  $Q$ ”).

We are now ready to formalize the idea of a  $d$ -view that is useful for  $Q$  but reveals as little as possible:

**Definition 3 (Minimally informative useful views).** Given a class of views  $\mathcal{C}$  and a query  $Q$ , we say that a  $d$ -view  $\mathcal{V}$  is a minimally informative useful  $d$ -view for  $Q$  within  $\mathcal{C}$  if  $\mathcal{V}$  is useful for  $Q$  and, for any other  $d$ -view  $\mathcal{V}'$  useful for  $Q$  based on views in  $\mathcal{C}$ ,  $\mathcal{V}'$  determines the view definition of each view in  $\mathcal{V}$ .

**Example 2.** Returning to Ex. 1, consider a query  $Q$  asking if there is a title and author name common to SIGMOD and VLDB, a possible double-submission:

$$\begin{aligned} &\exists \text{paperid title authorid authorname authorid' paperid'} \\ &\quad \text{VLDB}(\text{paperid}, \text{title}, \text{authorid}, \text{authorname}) \wedge \\ &\quad \text{SIGMOD}(\text{paperid}', \text{title}, \text{authorid}', \text{authorname}) \end{aligned}$$

Obviously there are many useful  $d$ -views for  $Q$ : we could export all information from each source, which be more than sufficient to answer  $Q$ . But intuitively there is an obvious candidate for the minimally informative useful view. From the VLDB source we should export

$$\begin{aligned} &\exists \text{authorid authorname} \\ &\quad \text{VLDB}(\text{paperid}, \text{title}, \text{authorid}, \text{authorname}) \end{aligned}$$

and similarly for SIGMOD. That is, to minimally support the query  $Q$  that joins across sources, we should project out everything but the variables that cross sources.

This is called the *canonical  $d$ -view* of  $Q$  with respect to the distributed schema. It will follow from the results mentioned in the next section that this really is the minimally-informative  $d$ -view for this example. From minimally-informativeness, it follows that if the interface designers want to support this double-submission query with *any*  $d$ -view, they will need to reveal all the paper titles submitted to each conference.  $\triangleleft$

All of these definitions can be additionally parameterized by integrity constraints  $\Sigma$ . Given  $d$ -instance  $\mathcal{D}$  satisfying  $\Sigma$  and a  $d$ -view  $\mathcal{V}$ ,  $\mathcal{V}$  *determines* a query  $Q$  over the  $d$ -schema at  $\mathcal{D}$  relative to  $\Sigma$  if: for every  $\mathcal{D}'$  *satisfying*  $\Sigma$  that is  $\mathcal{V}$ -indistinguishable from  $\mathcal{D}$ ,  $Q(\mathcal{D}') = Q(\mathcal{D})$ . We say that  $\mathcal{V}$  is *useful* for a query  $Q$  relative to  $\Sigma$  if it determines  $Q$  on every  $d$ -instance *satisfying*  $\Sigma$ . In the results in the next section we will assume that any integrity constraints are traditional database constraints, like Tuple-Generating Dependencies [14]. In the distributed schema setting, our positive results will usually assume that constraints are *local*: each constraint involves relations on a single source.

## 4. Results on interfaces via minimally informative views

We now present some results from [12] concerning when minimally informative views exist.

**Theorem 3.** *[Minimally informative d-views exist] [12] For any query  $Q$ , there are minimally informative d-views. The same holds in the presence of local constraints.*

The minimally informative d-view is not given constructively, but rather is defined via a Myhill-Nerode style equivalence, which we will discuss further in Section 5. Of course if  $Q$  is an arbitrary query, we cannot say much about what the minimally informative views are. But in the case where  $Q$  is a CQ, we can get a very simple minimally informative view.

**Theorem 4.** *[Minimally informative relational algebra d-views exist for CQs] [12] For any conjunctive query  $Q$ , there are minimally informative d-views, computable from  $Q$  and expressible in relational algebra. The same holds in the presence of local constraints.*

Notice here that we mentioned relational algebra, rather than CQ or UCQ. Surprisingly, we may need to use relational algebra features beyond UCQs:

**Theorem 5.** *[12] There are CQs  $Q$  where the minimally informative d-views are not CQs or even UCQs.*

*Thus in particular there are queries where the minimally informative d-view is not the canonical d-view.*

**Example 6.** Suppose we have two sources one with ternary relation  $R$ , the other with unary relation  $S$ . Consider the conjunctive query  $Q$ :

$$\exists x_1, x_2, y. R(x_1, x_2, y) \wedge R(y, x_2, x_1) \wedge S(x_1) \wedge S(x_2)$$

The canonical view at the  $R$  source is  $\{x_1, x_2 | \exists y. R(x_1, x_2, y) \wedge R(y, x_2, x_1)\}$  and at the  $S$  source it is  $\{x_1, x_2 | S(x_1) \wedge S(x_2)\}$ : these are the “obvious” views to support the query. But they are not the minimally informative views!

Consider the views formed by replacing the view on  $R$  with the view exporting the pairs  $x_1, x_2$  such that

$$(\exists y. R(x_1, x_2, y) \wedge R(y, x_2, x_1)) \vee (\exists y. R(x_2, x_1, y) \wedge R(y, x_1, x_2))$$

We reveal to an external user that either the pair  $x_1, x_2$  or its reverse satisfies a certain pattern. This is strictly less informative than the canonical views, which reveal which of the two cases holds. But the reader can check that it is sufficient to answer the query  $Q$ .  $\triangleleft$

What about d-views that are minimally informative *within the class of CQs*? These always exist as well, and it turns out that after minimizing  $Q$  (removing redundant conjuncts), they are indeed the canonical ones:

**Theorem 7 (Minimally informative CQ views exist).** *[12] Let  $Q$  a CQ that has no redundant conjuncts. Then the canonical d-view of  $Q$  is minimally informative useful within the class of CQ views. The same holds in the presence of local constraints.*

Finally, notice that in Theorem 3 we required any integrity constraints to be local. Arguably the simplest kind of non-local constraint is a *replication constraint*, saying that a relation  $R_1$  in source  $d_1$  is identical to a relation  $R_2$  in source  $d_2$ .

**Theorem 8.** [12] *There is a CQ  $Q$  and a single replication constraint where no minimally informative  $d$ -view exists.*

[12] shows that a weaker notion of minimally informative view exists in the presence of replication constraints, based on the “amount of disclosure”.

## 5. Indistinguishability relations

We now look at another way to define interfaces to data, based very roughly on work in [13]. An *indistinguishability relation* is an equivalence relation on instances of a schema. Clearly any kind of classical view gives an indistinguishability relation: two instances are indistinguishable if the views produce the same output. But it is natural to allow a data designer to define an interface by simply declaring which pairs of databases should be indistinguishable, and then let the system develop a method of querying – e.g. through translation to appropriate views. Indistinguishability relations play a role in [12], where often views are constructed implicitly via indistinguishability. For example, in the proof of Theorem 3 from the previous section, what one actually shows is:

**Theorem 9.** *For any query  $Q$  over a  $d$ -schema, there are minimally informative  $d$ -views, with each view given implicitly by an indistinguishability relation.*

This is what we referred to in the last section when we said that the minimally informative views are given by a Myhill-Nerode style equivalence relation. We sketch the argument, since it gives the idea of how to implicitly define a view via an indistinguishability relation. For a source  $s$  in a  $d$ -schema, an  $s$ -context is an instance for each source other than  $s$ . Given a source  $s$  and a query  $Q$ , we say two  $s$ -instances  $\mathcal{I}, \mathcal{I}'$  are  $(s, Q)$ -equivalent if for any  $s$ -context  $C$ ,

$$(\mathcal{I}, C) \models Q \Leftrightarrow (\mathcal{I}', C) \models Q$$

We say two  $d$ -instances  $\mathcal{D}$  and  $\mathcal{D}'$  are *globally  $Q$ -equivalent* if for each source  $s$ , the restrictions of  $\mathcal{D}$  and  $\mathcal{D}'$  over source  $s$ , are  $(s, Q)$ -equivalent.

Global  $Q$ -equivalence is clearly an indistinguishability relation. It is also not difficult to see that the corresponding  $d$ -view is a minimally informative useful  $d$ -view for  $Q$  within the class of all views:

Now let us return to indistinguishability relations in general. It is particularly interesting to look at indistinguishability relations defined by logical sentences. For a schema  $\mathcal{S}$ , let  $\mathcal{S}^2$  denote the vocabulary with two copies of each relation  $R$  in  $\mathcal{S}$ , denoted  $R$  and  $R'$ . Thus an instance for  $\mathcal{S}^2$  is a pair of instances for  $\mathcal{S}$ , one using the primed relations and one using the unprimed relations. We write such a pair as  $(\mathcal{I}_1, \mathcal{I}_2)$ . Thus  $\mathcal{S}^2$  is the natural vocabulary for describing relationships between structures.

**Definition 4.** Let  $\eta$  be a domain independent first-order logic sentence over  $\mathcal{S}^2$ . We say that  $\eta$  is a first-order logic indistinguishability relation if

- $\eta$  defines a transitive relationship on pairs of instances: for any instances  $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$   $(\mathcal{I}_1, \mathcal{I}_2) \models \eta$ , and  $(\mathcal{I}_2, \mathcal{I}_3) \models \eta$  implies  $(\mathcal{I}_1, \mathcal{I}_3) \models \eta$
- $\eta$  is commutative:  $(\mathcal{I}_1, \mathcal{I}_2) \models \eta$  if and only if  $(\mathcal{I}_2, \mathcal{I}_1) \models \eta$ .
- $\eta$  is reflexive: for all  $\mathcal{I}_1$ ,  $(\mathcal{I}_1, \mathcal{I}_1) \models \eta$

We can similarly relativize this to constraints  $\Sigma$  on  $\mathcal{S}$ :  $\eta$  need only define an equivalence relation on structures satisfying  $\Sigma$ . We can look at indistinguishability relations over all instances for  $\Sigma$ , or deal with sentences  $\eta$  that define an indistinguishability relation only on finite instances.

The results in [13] do not restrict to the finite case, and have several other deviations from the formalism as we present it here. But as we explain below, many of the results apply to the setting of finite instances.

### 5.1. Indistinguishability relations and relational/nested relational views

We mentioned that the obvious way to get an indistinguishability relation is via views. And if we use relational algebra views, we get a first-order logic indistinguishability relation. Let  $\Gamma = \{\gamma_1 \dots \gamma_n\}$  be a finite set of relational algebra views, or equivalently, active domain formulas. We let  $\eta_\Gamma$  be the  $\mathcal{I}^2$  sentence  $\bigwedge_i \forall \mathbf{x} [\gamma_i(\mathbf{x}) \leftrightarrow \gamma'_i(\mathbf{x})]$ . Here, for any domain-independent logical formula  $\varphi$  over  $\mathcal{S}$ ,  $\varphi'$  is the formula formed by priming every relation. For any such  $\Gamma$ ,  $\eta_\Gamma$  is a first-order logic indistinguishability relation. We call this a *relational algebra view indistinguishability relation*. We now describe a generalization of the notion of relational algebra view to *nested relations*. We refer here to the notion of nested set in database theory [16, 17, 18], and we can consider views given by the nested relational calculus. We will not need the exact definitions of nested relational calculus here. The observation is that agreement on nested relational views is expressible as a *first-order* logic sentence over  $\mathcal{S}^2$ : even though we are dealing with nested sets, we do not need “higher order logic” for the corresponding indistinguishability relation.

**Example 10.** Let schema  $\mathcal{S}$  consist of binary relation  $R$ . Consider the  $\mathcal{S}^2$  sentence:

$$\begin{aligned} &\forall x \exists x' \forall y. [R(x, y) \leftrightarrow R'(x', y)] \wedge \\ &\forall x' \exists x \forall y. [R(x, y) \leftrightarrow R'(x', y)] \end{aligned}$$

This is a first-order logic indistinguishability relation. It states that every adjacency set of an element in one instance is the adjacency set of a (possibly different) element in the other set. Informally, the family of adjacency sets in the instances are the same.

If schema  $\mathcal{S}$  consists of a ternary relation  $T(x, y, z)$ , in any instance  $\mathcal{I}$  containing  $x$  and  $y$ , we can let  $A_{\mathcal{I}}(x, y)$  be the  $z$ ’s such that  $T(x, y, z)$  holds in  $\mathcal{I}$ . We let  $A_{\mathcal{I}}(x)$  be the set of sets  $A_{\mathcal{I}}(x, y)$  as  $y$  ranges over nodes in  $\mathcal{I}$ , for a given  $x$  in  $\mathcal{I}$ . And we let  $A_{\mathcal{I}}$  be the set of sets of sets  $A_{\mathcal{I}}(x)$  as  $x$  ranges over nodes in  $\mathcal{I}$ . Then there is a first-order logic indistinguishability relation corresponding to preserving the nested set  $A_{\mathcal{I}}$ .  $\triangleleft$

In general, a *k-nested view indistinguishability relation* is given by a *k-partitioned formula*; this an active domain first-order formula in which the free variables are partitioned into *k* sets. We write such formulas  $\gamma(\mathbf{x}_1; \dots; \mathbf{x}_k)$ . In the binary relation example above, the partitioned formula is just  $R(x; y)$ , while in the ternary example it is  $T(x; y; z)$ . The definition of the nested set and the equivalence relation corresponding to  $\gamma(\mathbf{x}_1; \dots; \mathbf{x}_k)$  is given by induction on *k*, generalizing the example.

What kinds of first-order logic indistinguishability relations are there beyond nested relational views? Over infinite models, it is easy to arrive at logic-based indistinguishability relations that are not nested views.

**Example 11.** Let our vocabulary have binary relations  $<$  and let  $\Sigma$  state that  $<$  is a dense linear order. Let  $\eta$  be the sentence over  $\mathcal{S}^2$  stating:

$$\begin{aligned} & [\forall xyz. x < y < z \rightarrow \\ \exists uv. u <' y <' v \wedge (\forall q. u <' q <' v \rightarrow x < q < z)] \wedge \\ & [\forall xyz. x <' y <' z \rightarrow \\ \exists uv. u < y < v \wedge (\forall q. u < q < v \rightarrow x <' q <' z)] \end{aligned}$$

$\eta$  says that every open interval  $(x, y)$  in one dataset is the union of open intervals  $(u, v)$  in the other dataset, and vice versa. Then  $\eta$  is a logic-based indistinguishability relation. One can show that it is not a *k-nested view* for any *k*.

◁

Note that we made use of an infinite linear order here.

The following natural question is open:

**Question 1.** *Is there a set of first-order integrity constraints and a first-order indistinguishability relation that is not given by a k-nested relation view over finite structures?*

A negative answer would be a kind of semantics-to syntax or preservation theorem over finite structures, and only a few such results are known [19].

## 5.2. Some results

We present some of the results from [13] that could be interesting in the database context, warning the reader that the terminology and the formalism here does not match that in [13], which uses classical model theory as a framework.

It is not so difficult to show that the indistinguishability relations based on nested relational views form a strict hierarchy:

**Proposition 1.** [13] *For every  $n$  there are  $(n + 1)$  nested view indistinguishability relations that are not given by  $n$  nested view indistinguishability relations.*



For separation, we could use generalizations of the adjacency-based equivalence relation in Example 10: the set of adjacency sets of a binary relation, the set of sets of adjacency sets of a ternary relation, etc.

In [13] this is argued explicitly for general  $n$  only over infinite structures, but in the finite is illustrated in the case of 2 vs. 1. To prove the separation, one uses dense structures. In fact, one can show that this is necessary:

**Theorem 12.** [13] *If  $C$  is a class of sparse finite structures (e.g. graphs excluding a minor) then every equivalence relation given by a  $k$ -nested view indistinguishability relation is equivalent to one given by a relational algebra view indistinguishability relation.*

**Example 13.** Let us to the adjacency-based indistinguishability relation, the first one in Example 10. Suppose the datasources are restricted to be finite trees, where  $R$  is the child relation. We can see that the only way two trees can have the same adjacency sets is if they are identical! Thus, over trees, this equivalence relation is presented by a relational algebra view.  $\triangleleft$

If we consider the  $\mathcal{S}^2$  sentence corresponding to a relational algebra view indistinguishability relation, we only need a block of universal quantifiers that cross the two instances: it is of the form  $\forall x [\varphi(\mathbf{x}) \leftrightarrow \varphi'(\mathbf{x})]$ , where  $\varphi$  concerns only the unprimed instance and  $\varphi'$  the primed instance. Call such indistinguishability relations  $\Pi_1$ . In contrast, if we look at nested view indistinguishability, we see at least three quantifier blocks  $\forall \mathbf{x} \exists \mathbf{y} \forall \mathbf{z} (\dots)$ . In the first example within Example 10, the pattern was  $\forall x \exists x' \forall y$ . In the second example, which involved ternary relation  $T$  and sets of sets of sets, there would be an additional alternation of universal and existential. The following result (which is shown looking at indistinguishability over both finite and infinite structures), shows that the jump from one block to at least three blocks is essential:

**Theorem 14.** [13] *Suppose an indistinguishability relation is given by a  $\Pi_2$  sentence: of the form  $\forall x_1 \dots \exists y_1 \dots$ . Then it is given by a  $\Pi_1$  indistinguishability relation.*

Although the canonical example of  $\Pi_1$  relations are those given by relational algebra views, it is shown that there is an indistinguishability relation that is  $\Pi_1$  that is not given by relational algebra views. As with Question 1, we do not know if  $\Pi_1$  corresponds to relational algebra when only finite structures are considered.

## 6. Conclusion

The goal in this short article is to point to some work related to the expressiveness of mechanisms for interfacing with datasources. It is striking that the notion of interface design and the notion of indistinguishability appear so frequently in areas of computer science – see, for example [20] for a short survey on the latter – but relatively infrequently in the discussion of data management interfaces.

The particular formalizations we summarize are meant to be just examples of a broader set of questions. What are the different ways of defining datasource interfaces, and how can we compare their expressiveness? How can we answer queries against non-traditional interfaces like indistinguishability relations? What kind of disclosure vs. privacy trade-offs can we achieve with different kinds of interfaces?



## References

- [1] F. N. Afrati, R. Chirkova, *Answering Queries Using Views*, Second Edition, Morgan & Claypool Publishers, 2019.
- [2] M. Tork Roth, M. Arya, L. M. Haas, M. J. Carey, W. F. Cody, R. Fagin, P. M. Schwarz, J. T. II, E. L. Wimmers, The Garlic Project, in: SIGMOD, 1996.
- [3] C. Li, Computing complete answers to queries in the presence of limited access patterns, VLDB Journal 12 (2003) 211–227.
- [4] C. Li, E. Chang, Query planning with limited source capabilities, in: ICDE, 2000.
- [5] A. Nash, B. Ludäscher, Processing union of conjunctive queries with negation under limited access patterns, in: EDBT, 2004.
- [6] J. D. Ullman, *Principles of Database and Knowledge-Base Systems*, V2, Comp. Sci. Press, 1989.
- [7] A. Rajaraman, Y. Sagiv, J. D. Ullman, Answering queries using templates with binding patterns, in: PODS, 1995.
- [8] A. Deutsch, B. Ludäscher, A. Nash, Rewriting queries using views with access patterns under integrity constraints, Theor. Comput. Sci. 371 (2007) 200–226.
- [9] J. Romero, N. Preda, A. Amarilli, F. M. Suchanek, Equivalent rewritings on path views with binding patterns, in: ESWC, 2020.
- [10] A. Amarilli, M. Benedikt, When can we answer queries using result-bounded data interfaces?, Log. Methods Comput. Sci. 18 (2022).
- [11] B. Cautis, A. Deutsch, N. Onose, Querying data sources that export infinite sets of views, Theory Comput. Syst. 49 (2011) 367–428.
- [12] M. Benedikt, P. Bourhis, L. Jachiet, E. Tsamoura, Balancing expressiveness and inexpressiveness in view design, TODS (2021).
- [13] M. Benedikt, E. Hrushovski, Model equivalences, 2024. [Arxiv.org/pdf/2406.15235](https://arxiv.org/pdf/2406.15235).
- [14] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.
- [15] A. Nash, L. Segoufin, V. Vianu, Views and queries: Determinacy and rewriting, TODS 35 (2010).
- [16] M. Gyssens, D. V. Gucht, The expressiveness of query languages for nested relations, IEEE Data Eng. Bull. 11 (1988) 48–55.
- [17] J. Paredaens, D. V. Gucht, Possibilities and limitations of using flat operators in nested algebra expressions, in: PODS, 1988.
- [18] P. Buneman, S. Naqvi, V. Tannen, L. Wong, Principles of Programming with Complex Objects and Collection Types, Theoretical Computer Science 149 (1995) 3–48.
- [19] B. Rossman, Homomorphism preservation theorems, J. ACM 55 (2008).
- [20] H. Attiya, S. Rajsbaum, Indistinguishability, Commun. ACM 63 (2020) 90–99.