BAYESSTORE: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models

Daisy Zhe Wang[*], Eirinaios Michelakis[*], Minos Garofalakis[†*], and Joseph M. Hellerstein[*]
[*] Univ. of California, Berkeley EECS and [†] Yahoo! Research

ABSTRACT

Several real-world applications need to effectively manage and reason about large amounts of data that are inherently uncertain. For instance, pervasive computing applications must constantly reason about volumes of noisy sensory readings for a variety of reasons, including motion prediction and human behavior modeling. Such probabilistic data analyses require sophisticated machine-learning tools that can effectively model the complex spatio/temporal correlation patterns present in uncertain sensory data. Unfortunately, to date, most existing approaches to probabilistic database systems have relied on somewhat simplistic models of uncertainty that can be easily mapped onto existing relational architectures: Probabilistic information is typically associated with individual data tuples, with only limited or no support for effectively capturing and reasoning about complex data correlations. In this paper, we introduce BAYESSTORE, a novel probabilistic data management architecture built on the principle of handling statistical models and probabilistic inference tools as first-class citizens of the database system. Adopting a machine-learning view, BAYESSTORE employs concise statistical relational models to effectively encode the correlation patterns between uncertain data, and promotes probabilistic inference and statistical model manipulation as part of the standard DBMS operator repertoire to support efficient and sound query processing. We present BAYESSTORE's uncertainty model based on a novel, *first-order statistical model*, and we redefine traditional query processing operators, to manipulate the *data* and the *probabilistic models* of the database in an efficient manner. Finally, we validate our approach, by demonstrating the value of exploiting data correlations during query processing, and by evaluating a number of optimizations which significantly accelerate query processing.

1 Introduction

There is growing acknowledgment among database researchers and practitioners that modern database systems need to routinely deal with large amounts of *uncertain* information, be it incorrect, incomplete, or internally inconsistent. Work on *Probabilistic Database Systems (PDBSs)* has the goal of addressing this problem with techniques to help quantify, explain, and manage uncertainty — all within the familiar context of relational database models and languages, and without sacrificing scalability over the stored data collections.

PVLDB '08, August 23-28, 2008, Auckland, New Zealand

Of course, the fundamental mathematical tools for managing uncertainty come from probability and statistics. In recent years, these tools have been aggressively imported into the computational domain under the rubric of *Statistical Machine Learning (SML)*. Of special note here is the widespread use of *Graphical Modeling* techniques, including the many variants of Bayesian Networks (BNs) and Markov Random Fields (MRFs) [13]. These techniques can provide robust statistical models that capture complex *correlation patterns* among variables, while, at the same time, addressing some computational efficiency and scalability issues as well. Graphical models have been applied with great success in applications as diverse as signal processing, information retrieval, sensornets and pervasive computing, robotics, natural language processing, and computer vision.

Recent research efforts in PDBSs have injected new excitement into the area of uncertainty management in database systems. Unfortunately, the bulk of this work has, to date, relied on somewhat simplistic models of uncertainty, placing the focus on simple probabilistic extensions that can be easily mapped to existing relational database architectures, and essentially ignoring the state-of-the-art in SML. For instance, existing PDBSs typically associate probabilities directly with data at the level of individual *tuples* or *tuple values*. While such fine-grained probabilistic information may be warranted in certain scenarios (e.g., data integration), they also often give rise to intractably large probabilistic reasoning problems [7]. Furthermore, in several application domains, including pervasive computing and sensornets, the *granularity of uncertainty* can be much coarser depending on the underlying random process being observed (e.g., all readings from sensor-1 follow the same distribution pattern). Existing PDBSs also offer very limited or no support for effectively modeling and reasoning about complex correlation patterns — unfortunately, as SML work demonstrates, such correlation patterns abound in real-world data. In short, existing PDBSs simply cannot support realistic, state-of-the-art probabilistic reasoning within the database system: Such reasoning currently needs to occur *outside the database* and its results can only be approximately mapped to and stored within the simplified uncertainty models supported by the PDBS; see, for instance, [11] for such an approximate mapping in the context of MRF-based information extraction.

Related Work. While traditional SML has provided well-founded mathematical tools for uncertainty management, such tools are not targeted at the declarative management and processing of large-scale data sets. Since the early 80's, a number of PDBSs have been proposed in an effort to address this issue [12, 5, 3, 10, 7, 4, 17, 2, 1]. Moving away from statistical approaches, this work extends the relational model with probabilistic information captured at the level of individual *tuple existence* (i.e., a tuple may or may not ex-

ist in the DB) [5, 7, 10, 17] or individual *tuple-value uncertainty* (i.e., an attribute value in a tuple follows a probabilistic distribution) [3, 4, 2, 1]. The Trio [4] and MayBMS [2, 1] efforts, in particular, try to adopt both types of uncertainty, with Trio focusing on promoting *data lineage* as a first-class citizen in PDBSs and MayBMS aiming at more efficient tuple-level uncertainty representations through effective relational table decompositions. In all cases, probabilities are directly associated (and, stored) with individual tuples and/or tuple values and are processed using standard relational query operators over uncertain tables — this is another major departure from SML, that typically imposes a clear separation between observed data (i.e., evidence) and uncertainty models (e.g., BNs or MRFs) [13].

Query processing in PDBSs is typically based on the standard *possible worlds semantics*, where a PDB is viewed as encoding a probability distribution over all possible deterministic instances. As demonstrated by Dalvi and Suciu [7], such query processing quickly gives rise to *computationally-intractable* probabilistic inference problems, as complex correlation patterns can emerge during processing even if naive independence assumptions are made on the base data tables. In fact, modulo a restricted class of "safe" query execution plans, query processing in tuple-uncertain PDBSs is $\#P$-complete *in the size of the database* [7]. This clearly raises some serious practicality concerns for PDBSs, which, in our view, are largely due to the *intractably fine granularity* of uncertainty modeling in current PDBS architectures. The same is true for more recent proposals that employ either (a) BNs with tuple-existence random variables to model tuple correlations in the possible-worlds distribution [17] or (b) schema decompositions and tuple random variables to *factor* the possible-worlds distribution [2, 1]: The size of the underlying probability model (e.g., tuple-existence BN) is *linear in the size of the database*, raising serious scalability concerns for complex query processing and probabilistic reasoning over probabilistic data.

At the same time, as exemplified by recent research on *statistical relational learning*, the SML community has also been moving in the direction of higher-level, more declarative probabilistic modeling tools. In a nutshell, the key idea lies in learning and reasoning with *First-Order (FO)* (or, *relational*) probabilistic models, where the random variables in the model representation correspond to *sets of random parameters* in the underlying data [9, 15, 16]. *Probabilistic Relational Models (PRMs)* [9] are a good example of such models, formed as a FO extension of traditional (propositional) BNs over a relational DB schema: Random variables in the PRM directly correspond to attributes in the schema; thus, PRMs can capture correlation structure across attributes of the same or different relations (through foreign-key paths) *at the schema level*. The idea, of course, is that this correlation structure is *shared* (or, FO-quantified) across all data tuples in the relation(s); for instance, in a `Person` database, a PRM can specify a correlation pattern between `Person.Weight` and `Person.Height` (e.g., a conditional probability distribution $\mathsf{Pr}[\texttt{Person.Weight} = x | \texttt{Person.Height} = y]$ that is shared across all `Person` instances. Such shared correlation structures not only allow for *more concise and intuitive* statistical models, but also enable the use of more efficient *lifted probabilistic inference* techniques [16] that work directly off the concise FO model.

Our Contributions. In this paper, we present a model and initial implementation of a novel PDBS we call BAYESSTORE, which treats rich graphical models as first class objects, alongside a traditional relational storage. Building on seminal work incorporating graphical models and relations [9, 8, 17], we have developed a scalable and statistically robust PDBS, with a number of key distinguishing features:

- A new data uncertainty model based on a set of novel *First-Order (FO)* extensions to graphical models, that enable declarative specifications of both tuple, and attribute level correlations, among *populations* of data items.[1]
- Seamless integration of state-of-the-art SML techniques with relational query processing, to directly support both relational queries and probabilistic model reasoning and manipulations, inside the PDBS.
- Query optimizations able to provide significant performance benefits, by exploiting graphical models to filter out unlikely tuples, without impairing the soundness or accuracy of the result.

In this paper we describe the BAYESSTORE system, its current state of implementation, and initial experiments demonstrating the importance of our approach from the perspectives of both performance and statistical robustness.

2 The BAYESSTORE Data Model

BAYESSTORE is founded on a novel data model that treats (uncertain) relational data and statistical models of uncertainty as first-class citizens of the PDBS. In this section, we outline the system's data model, focusing, in particular, on a *novel, declarative FO extension of* BN *models* that is able to capture complex possible-worlds distributions of large PDB instances in a compact and scalable manner. Based on the solid probabilistic foundation of BNs, BAYESSTORE can model the complex correlation patterns present in real-world uncertain applications; in addition, through our novel FO extensions, such correlations can be declaratively expressed (and learned) *at the appropriate level of granularity*, with random variables (RVs) specified in a very flexible and schema-independent manner.[2] (This is in contrast to PRMs, where RVs correspond only to schema-level attributes; PRMs are a simple special case of the BAYESSTORE FO statistical model.)

2.1 Incomplete Relations and Possible Worlds.

Abstractly, a BAYESSTORE probabilistic database $\mathcal{DB}^p = <\mathcal{R}, \mathrm{F}>$ consists of two key components: (1) A collection of *incomplete relations* $\mathcal{R}$, and (2) A *probability distribution function* F that quantifies the uncertainty associated with all incomplete relations in $\mathcal{R}$.

An incomplete relation $R \in \mathcal{R}$ is defined over a schema $\mathcal{A}^d \cup \mathcal{A}^p$ comprising a (non-empty) subset $\mathcal{A}^d$ of *deterministic attributes* (that includes all candidate and foreign key attributes in R), and a subset $\mathcal{A}^p$ of *probabilistic attributes*. Deterministic attributes have no uncertainty associated with any of their values — they always contain a legal value from their domain, or the traditional SQL NULL. On the other hand, the values of probabilistic attributes may be present or *missing* from R. Given a tuple $t \in$R, non-missing values for probabilistic attributes are considered *evidence*, representing our partial knowledge of t. Missing values for probabilistic attribute $A_i \in \mathcal{A}^p$ capture attribute-level uncertainty; formally, each such missing value is missing value is associated with a RV X_j ranging over $\mathsf{dom}(A_i)$ (the domain of attribute A_i). (Note that applications requiring *tuple-existence uncertainty* can incorporate

[1] In a recent workshop paper, Sen et al. [18] also discuss the use of FO probabilistic models in PDBSs. Still, their FO model is quite different from ours; furthermore, the interaction of relational query processing and FO model reasoning is not explored in their paper.

[2] While our focus here is on *directed* graphical models (i.e., BNs), our key ideas are also applicable to the undirected case; due to space constraints, details are deferred to the full paper.

a probabilistic boolean attribute $\texttt{Exist}^p$ to capture the uncertainty of each tuple's *existence* in an incomplete relation. As discussed later, this requirement can also arise during query processing in BAYESSTORE.)

The second component of a BAYESSTORE database $\mathcal{DB}^p$=<$\mathcal{R}$, F > is a probability distribution function F, that models the *joint distribution* of all missing-value RVs for relations in $\mathcal{R}$. Thus, assuming n such RVs, $\mathrm{X}_1, \ldots, \mathrm{X}_n$, F denotes the joint probability distribution $\mathsf{Pr}(\mathrm{X}_1, \ldots, \mathrm{X}_n)$. The association of F with conventional PDB *possible- worlds semantics* [7] is now straightforward: every complete assignment of values to all X_i's maps to a *single* possible world for $\mathcal{DB}^p$.

Note that, unlike most PDB work to date [5, 10, 7, 4, 2], the BAYESSTORE data model employs a clean separation of the relational and probabilistic components: Rather than attaching probabilities to uncertain database elements, BAYESSTORE maintains the probability distribution F over the incomplete relations as a separate entity. This design has a number of benefits. First, it allows us to expose all probabilistic information as a separate, first-class database object that can be queried and manipulated by users/-applications. Second, and perhaps most important, it allows us to leverage ideas from state-of-the-art SML techniques in order to effectively *represent, query, and manipulate* the joint probability distribution F. Thus, the BAYESSTORE data model directly captures the PDB possible-worlds semantics, while exposing the (potentially complex) probabilistic and correlation structures in the data as as first-class objects that can be effectively modeled and manipulated using SML tools.

From our definition of F, it is evident that it is a potentially huge mathematical object — the straightforward method for representing and storing this n-dimensional distribution F is essentially linear in the number of possible worlds of $\mathcal{DB}^p$. We discuss two representation techniques that help capture this information far more compactly. First, we describe *Bayesian Networks* (BNs), a traditional SML tool that exploits *conditional independence* of RVs to obtain more efficient, factored representations of joint distributions. Then, we discuss a set of novel FO extensions to BN models that enable BAYESSTORE to obtain even more compact probabilistic model representations through a flexible, declarative model for specifying *shared probabilistic and correlation structures*. We first take some time to introduce a simple example that will be used to illustrate some key concepts in what follows.

EXAMPLE 1. *Suppose there are two sets of environmental sensors in different rooms of a hotel. The first set of sensors monitors temperature and light levels; while the second monitors humidity and light levels. Sensor readings are inherently noisy – readings may be dropped due to sensor malfunctions or power fluctuations, or even garbled by radio interference from other equipment. For the ease of exposition, we assume that sensor readings are discretized into binary values; $Cold$ and Hot for temperature, Drk (Dark) and Brt (Bright) for light, and $High$ and Low for humidity levels. We regard all collected readings from the two sets of sensors being stored in a database $\mathcal{DB}^p$ of two incomplete relations: Sensor1(Time(T), Room(R), Sid, temp^p(Tp), light^p(L)), and Sensor2(Time(T), Room(R), humidity^p(H), light^p(L)), respectively. Probabilistic attributes are denoted with superscript p, while the attributes constituting the primary key are underlined. In this example, neither relation is associated with tuple-level existence uncertainty. Figure 1(a) depicts an instance of these relations. Blank cells correspond to missing values, and are represented by a* RV *for each, taking values from the domain of the corresponding probabilistic attribute. Each* RV *can be referred to by its tuple identifier and attribute name (e.g. $\varepsilon_1.Tp$ is the* RV *of the missing value of tuple ε_1). In this snapshot of $\mathcal{DB}^p$, the possible worlds distribution F can be captured by an 8-dimensional distribution table, over the 8* RV*s associated with the uncertain quantities of the relations.*

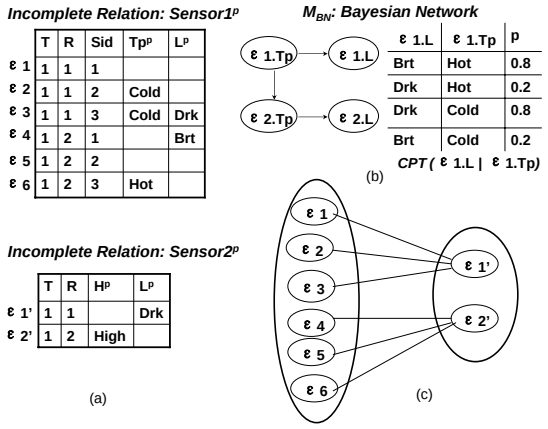


Figure 1: (a) Incomplete Relations *Sensor1* and *Sensor2*. Each tuple is referred to by its identifier (ε_i). (b) An example Bayesian Network representation of the probabilistic distribution of tuples $\{\varepsilon_1, \varepsilon_2\}$, and the CPT associated with RV $\varepsilon_1.L$. (c) The mapping between the entity sets of two stripes, S_{T_p} and S_H, from *Sensor1* and *Sensor2* respectively, involved in a child-parent relationship.

2.2 Bayesian Networks: A Quick Primer

Over the past 20 years, *Bayesian Networks* (BNs) have emerged as a widely-used SML model to efficiently represent and reason about multi-variate probabilistic distributions [13]. A BN is a directed acyclic graph $G = (V, E)$ with nodes representing a set of RVs $\{X_1, X_2, \ldots, X_{|V|}\}$, and edges denoting conditional dependence relationships between RVs. In a nutshell, the structure of the BN encodes the assertion that *each node is conditionally independent of its non-descendants, given its parents*. This allows us to model the probability distribution of a RV X_i, *locally*, as a *Conditional Probability Table (CPT)* or *factor* $\mathsf{Pr}[X_i|Pa(X_i)]$ that specifies how X_i depends probabilistically on the values of its parent nodes $Pa(X_i)$. It is not difficult to see that the overall joint probability distribution can be factored into the product of all local models in the DAG; that is, $\mathsf{Pr}[X_1, ..., X_{|V|}] = \prod_{i=1}^{|V|} \mathsf{Pr}[X_i|Pa(X_i)]$. Given that the dimensionality of local correlation models is typically much smaller than $|V|$, this factored BN representation can be orders of magnitude more efficient than a straightforward $|V|$-dimensional probability distribution.

Thus, a natural, first approach to compress the multi-variate distribution F of a BAYESSTORE database $\mathcal{DB}^p$, is to encode it using a BN: Every probabilistic attribute value in $\mathcal{DB}^p$ is represented by one RV in the BN, and local CPT models capture probabilistic correlations across these values. Here, note that we are slightly generalizing our earlier description of the BAYESSTORE probabilistic model by associating a RV with *every* probabilistic attribute value, either missing or not. Such a generalized model essentially acknowledges the potential for uncertainty in all probabilistic values, and is closer to the traditional SML point of view where models for all random quantities are learned from historical training quantities and further manipulated on the basis of current observations. In addition, note that the required possible-worlds distribution function F over just missing-value RVs can be obtained using conventional

SML techniques for *conditioning* the BN on the observed *evidence values* provided in the incomplete relations[3].

Finally, we make a conceptual distinction between two types of correlations (i.e., edges in the BN). A correlation involving RVs mapping to attribute values *in the same tuple*, is termed a *horizontal correlation*; otherwise (i.e., if the correlation associates RVs from distinct tuples), it is termed a *vertical correlation*. As we will see, the distinction between the two is sometimes important during query processing in BAYESSTORE.

EXAMPLE 2. *Figure 1(b) depicts a simple* BN*, representing the joint* $\mathsf{Pr}[\varepsilon_1.Tp, \varepsilon_1.L, \varepsilon_2.Tp, \varepsilon_2.L]$ *of the 4* RV*s that correspond to the probabilistic values of tuples* ε_1 *and* ε_2*. The local model for each* RV *is given by a conditional probability table (CPT). The figure shows only the CPT of* RV $\varepsilon_1.L$*. The* RV $\varepsilon_2.Tp$ *is considered an* observed RV*, as its value is fixed to* $Cold$*, from the evidence provided in* Sensor1*. The joint distribution* F *of* Sensor1 *can be computed by calculating the probability* $\mathsf{Pr}[\varepsilon_1.Tp, \varepsilon_2.Tp, \varepsilon_2.L| \varepsilon_1.L = Cold]$ *over the* BN*. The edge between* $\varepsilon_1.Tp$ *and* $\varepsilon_2.Tp$ *constitutes a vertical correlation; the rest are horizontal.*

2.3 The BAYESSTORE FO Bayesian Network Model

Even with the use of a factored representation that exploits conditional independence among tuple variables, the size of the resulting tuple-based BN model (also referred to as a *propositional* BN) remains problematic: The model requires one RV per probabilistic attribute, per tuple, per relation! Such huge propositional BN models are clearly not useful as an intuitive representation of the uncertainty in the data; furthermore, the cost of probabilistic reasoning over these models is bound to be prohibitive for non trivially-sized domains [17].

To overcome these problems, BAYESSTORE employs a novel refinement of recent ideas in the area of *First-Order (FO)* probabilistic models [9, 15, 16]. More specifically, the BAYESSTORE probabilistic model is based on a novel class of *First-Order Bayesian Network (FOBN)* models. In general, a FOBN model $\mathcal{M}_{FOBN}$ extends traditional (propositional) BN structures by capturing independence relationships between *sets* of RVs. This allows correlation structures that are *shared* across multiple propositional RVs to be captured in a concise manner. More formally, a $\mathcal{M}_{FOBN}$ encodes a joint probability distribution by a set of *FO factors* $\mathcal{M}_{FOBN} = \{\phi_1, ..., \phi_n\}$ — each such factor ϕ_i represents the local CPT model of a *population* of (propositional) RVs. For instance, the popular *Probabilistic Relational Models (PRMs)* [9] are an instance of FOBN models specified over a given relational schema: First-order RVs in a PRM directly correspond to schema attributes, and the correlation structure specified is shared across *all propositional instances* in the relation (which are assumed to be independent across tuples — i.e., no vertical correlations). Attributes in different relation schemas can also be correlated along key/foreign-key paths [9].

The BAYESSTORE FOBN model is based on a non-trivial extension to PRMs that allows correlation structure to be specified and shared across first-order RVs corresponding to arbitrary, declaratively specified, sub-populations of tuples in a relational schema. Thus, unlike PRMs, shared correlation structures in BAYESSTORE can be specified in a very flexible manner, without being tied to a specific relational schema design. This flexibility can, in turn, enable concise, intuitive representations of probabilistic information. The key, of course, is to ensure that such flexibly-defined FO correlation structures are also guaranteed to map to a *valid probabilistic model* over the base (propositional) RVs in the BAYESSTORE database. (Such a mapping is known as *FO model "grounding"* in the SML literature.) In what follows, we give a brief overview of some of the key concepts underlying the BAYESSTORE FOBN uncertainty model, and how they ensure a valid mapping to a possible-worlds distribution.

First-Order RVs in BAYESSTORE: Entities, Entity Sets, and Stripes. The following definition provides a declarative means of specifying arbitrary sub-populations of tuples by selecting "slices" of the deterministic part of an incomplete relation R.

DEFINITION 2.1. *[Entity Set* $R.\mathcal{E}$ *] An Entity Set (*$R.\mathcal{E}$*) of an incomplete relation* R *with deterministic attributes* $K_1, \ldots, K_n$*, is the deterministic relation* $R.\mathcal{E}$ $(\mathcal{K}')$ *(where* $\mathcal{K}' \subseteq \{K_1, \ldots, K_n\}$*), defined as a relational algebra query which performs a selection and a subsequent projection on any part* $\mathcal{K}'$ *of* $\{K_1, \ldots, K_n\}$*:* $R.\mathcal{E} \stackrel{def}{=} \pi_{\mathcal{K}'}(\sigma_{condition\ on\ \mathcal{K}'}(\ R\ ))$.

The elements of an entity set essentially identify a number of "entities" at arbitrary granularities – tuples or tuple *groups* – which contain some probabilistic attributes that we intend to model. Note that, in general, since entity sets are defined as *projections* on parts of R's superkey, a particular entity ε can correspond to a *set of tuples* in R. The probabilistic attribute RV associated with ε corresponds, in fact, to *the same* RV across all R tuples corresponding to ε; in other words, in any possible world, a probabilistic attribute associated with an entity is *instantiated to the same value* for all tuples in the entity's extent. Thus, intuitively, entities define the granularity of the base, propositional RVs in the BAYESSTORE model. An *entity set* then naturally specifies the granularity of *first-order* RVs (quantified over all entities in the set).

Various entity sets can be specified for the same relation, depending on the level in which entities are captured. Among those, the *maximal* entity set of R, which associates an entity *with each tuple* in R, can be obtained by projecting on any key of R— for our *Sensor1* table example: *Sensor1*$.\mathcal{E}_M = \pi_{\{R,T,Sid\}}$(*Sensor1*). Each entity ε in *Sensor1*$.\mathcal{E}_M$ (i.e. each sensor reading tuple) is associated with two uncertain quantities, its temperature ($\varepsilon.Tp$) and light readings ($\varepsilon.L$), which can be represented by the RVs $\varepsilon.X_{Tp}$ and $\varepsilon.X_L$, that model the stochastic process of assigning values to $\varepsilon.Tp$ and $\varepsilon.L$, respectively.

Coarser entity definitions are also useful when we need to associate the same RV for several tuples in an incomplete relation, i.e., tuples that always have the same probabilistic attribute value in any possible-world instantiation. Such situations arise naturally, for instance, when values of a probabilistic attribute are redundantly repeated in a table or during the processing of cross-product and join operations over probabilistic attributes (Section 3.3).

Having specified the concept of entity sets, we can now naturally define the notion of first-order RVs in BAYESSTORE. Such FO RVs (termed *stripes*) represent the values of a probabilistic attribute for a *population of entities* that share the same probabilistic characteristics.

DEFINITION 2.2. *[Stripe* $S <R.\mathcal{A}^p, \mathcal{E} >$*] A stripe* $S <R.\mathcal{A}^p, \mathcal{E} >$ *over an entity set* $\mathcal{E}$ *of an incomplete relation* R*, represents a set of random variables, one per entity from* $\mathcal{E}$*, which models the stochastic process of assigning values to the probabilistic attribute* $\mathcal{A}^p$ *of that entity.*

First-Order Factors and FOBNs in BAYESSTORE. As mentioned earlier, a FO factor in a FOBN captures the shared cor-

[3] BNs can also accommodate continuous, or combinations of continuous and discrete RVs. In the interest of space, we restrict our attention to discrete distributions, and refer the interested reader to standard textbooks on the subject [6].

relation structure (CPT) for a group of underlying propositional RVs (i.e., entities). Using our earlier definitions, we can define BAYESSTORE *FO factor* $\phi\ (S|Pa(S))$, as a local CPT model that describes the conditional dependence relationship between a *child stripe* S and its *parent stripes* $Pa(S)$. For example, if for all entities in of the incomplete relation *Sensor1*, the RV corresponding to light, L_i, conditionally depends on the temperature, Tp_i, of the same entity ε_i (i.e., $Tp_i \rightarrow L_i$), and this correlation pattern (i.e., CPT) is shared across all entities, then this situation can be captured concisely by two stripes: $S_{Tp} =< Sensor.Tp, \{\pi_{R,T,Sid}Sensor\} >$ and $S_L =< Sensor.L, \{\pi_{R,T,Sid}Sensor\} >$, and a FO factor over them: $\phi\ (S_L|S_{Tp})$.

In the above example, we implicitly assumed the existence of a one-to-one mapping between entities in the child stripe (S_L) and entities in its parent stripe (S_{Tp}); i.e., each (propositional) RV $L_i \in S_L$ will have as a parent its corresponding $Tp_i \in S_{Tp}$. Although this is often the case, non-bijective types of stripe mappings can be defined as well.

DEFINITION 2.3. *[Mapping between Stripes f [S_p, S_c]] A mapping f between two stripes – a child stripe $S_c < R.\mathcal{A}_c^p, \mathcal{E}_c >$ and a parent stripe $S_p < R.\mathcal{A}_p^p, \mathcal{E}_p >$, is a surjective function from the entity set $\mathcal{E}_c$ to $\mathcal{E}_p$, $f[S_p, S_c]$: $\mathcal{E}_c \rightarrow \mathcal{E}_p$, expressed as a first-order logic formula over the schemata $\mathcal{K}_c$ and $\mathcal{K}_p$ of the entity sets $\mathcal{E}_c$ and $\mathcal{E}_p$, where $\mathcal{K}_p \subseteq \mathcal{K}_c$. For the mapping to be valid, the selection predicates of the relational algebra expressions used to define $\mathcal{E}_c$ and $\mathcal{E}_p$ respectively, need to be the same.*

Definition 2.3 formalizes the association of the individual elements (RVs) of two stripes involved in a "child-parent" relationship ($S_p \rightarrow S_c$) at a per-instance level. In essence, it requires that every RV from S_c will have a *single* corresponding parent RV from S_p. In addition, it ensures the minimality of the parent stripe S_p, through the constraint of surjectivity, as every RV in S_p has to be associated with *at least* one RV from S_c. From the above, while it is permissible for an RV from S_p to be parent of more than one RV from S_c, a child RV from S_c cannot have more than one parent (since such multi-variable correlations cannot be captured by a single FO edge).

The last requirement of definition 2.3 essentially states that between the entity set attributes of the child and the parent stripes, there is a key-foreign key relationship, so that the entities between the two sets can be unambiguously associated with each other. To continue the previous example, the "1-1" mapping between the stripe S_L and its parent S_{Tp}, can be expressed as:

$$(\forall \varepsilon_L \in S_L.\mathcal{E}, \forall \varepsilon_{Tp} \in S_{Tp}.\mathcal{E}) Pa(r.\varepsilon_L) = r.\varepsilon_{Tp},\ \text{s.t.:}\\ \varepsilon_L.T = \varepsilon_{Tp}.T \wedge \varepsilon_L.R = \varepsilon_{Tp}.R \wedge \varepsilon_L.Sid = \varepsilon_{Tp}.Sid \quad (1)$$

where $r.\varepsilon_L$ signifies the RV associated with the entity ε_L from stripe S_L's entity set, and $Pa(r.\varepsilon_L)$ the parent RV of $r.\varepsilon_L$.

Up to this point we have tacitly assumed that we can express correlations only among stripes of the same incomplete relation. Nevertheless, conditional dependence relationships *can* be formed between stripes of different relations. As noted earlier, the two relations must be involved in a key-foreign key relationship, the foreign key of the parent incomplete relation should be part of the primary key of the child, and the selection predicates (if any), populating the entity sets of the stripes must be the same, for the last condition of definition 2.3 to be satisfied.

Following the running example of Figure 1, we assume that attributes T and R of *Sensor* form a foreign key for *Sensor2*. Moreover, assume that for each room and timestamp, the temperature readings of the 3 different sensors in that room depend conditionally on the (single) humidity reading for that timestamp (i.e. $H \rightarrow Tp$), and that this correlation pattern is quantified by the same CPT across all such groups of RVs. That can be expressed by defining 2 stripes, $S_{Tp} =< Sensor.Tp, \{\pi_{R,T,Sid}Sensor\} >$ for temperature and $S_H =< Sensor2.H, \{\pi_{R,T}Sensor2\} >$ for humidity, and a first order factor $\phi(S_{Tp}|S_H)$ which is made explicit through the one-to-many mapping:

$$(\forall \varepsilon_{Tp} \in S_{Tp}.\mathcal{E}, \forall \varepsilon_H \in S_H.\mathcal{E}) Pa(r.\varepsilon_{Tp}) = r.\varepsilon_H,\ \text{s.t.}\\ \varepsilon_{Tp}.T = \varepsilon_H.T \wedge \varepsilon_{Tp}.R = \varepsilon_H.R \quad (2)$$

The mapping $f[S_{Tp}, S_H]$ is graphically depicted in Figure 1(c). In this example we can see the application of Definition 2.3 in its full generality; each entity of the parent stripe S_H is associated with its 3 corresponding entities from the child stripe S_{Tp}, which can be uniquely defined because of the key-foreign key relationships that characterize the two entity sets.

We are now ready to formally define a first-order factor.

DEFINITION 2.4. *[First-order Factor ϕ ($\{S_i\}$,$\{f_i\}$,CPT)] A first-order factor ϕ represents the conditional dependency of a child stripe S_c on a set of parent stripes $Pa(S_c) = \{S_{p_1}, \ldots, S_{p_n}\}$, $\phi\ (S_c|\pi(S_c))$. It consists of:*

- *A child stripe S_c and a (possibly empty) ordered list of parent stripes $\{S_{p_1}, \ldots, S_{p_n}\}$.*
- *A (possibly empty) ordered list of stripe mappings $\{f_i[S_c, S_{p_i}]\}$, $(i = 1, \ldots, n)$, each one associating the entities of the child stripe with those of the corresponding parent stripe.*
- *A conditional probability table (CPT), quantifying the common local model, that holds for all the RV associations that are defined by the set of mappings $\{f_i[S_c, S_{p_i}]\}$, among the entities of the stripes involved, $P(S_c|\pi(S_c))$.*

A BAYESSTORE FOBN model $\mathcal{M}_{FOBN}$ can now be defined as a set of first-order factors $\Phi = \{\phi_1, ..., \phi_n\}$, where each ϕ_i defines a local CPT model for the corresponding child stripe. Note that our definitions of stripes and stripe mappings are sufficient to guarantee that each individual FO factor can be grounded to a valid collection of local CPTs over propositional RVs (entities). In the presence of multiple FO factors (and possible connections across factors), ensuring that the global $\mathcal{M}_{FOBN}$ model can be grounded to a valid probabilistic distribution over entities is a little trickier. More specifically, note that even distinct stripes in factors can overlap (see Example 3 below), and additional structural constraints must be imposed on the model to guarantee grounding to an *acyclic* BN model. Due to space constraints, details are deferred to the full version of this paper.

EXAMPLE 3. *Figure 2 shows an example $\mathcal{M}_{FOBN}$ for our sensornet probabilistic database. There are three first-order factors:*

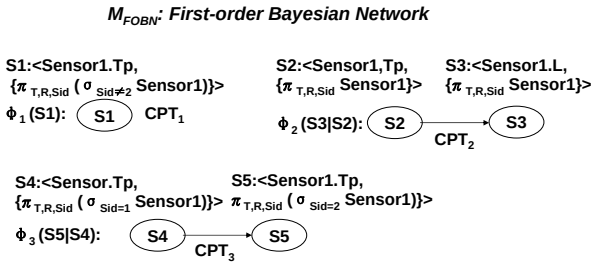


Figure 2: First-order Bayesian Network model over the incomplete relation *Sensor1* in Figure 1.

$\{\phi_1, \phi_2, \phi_3\}$. *For ϕ_1, the child stripe $\phi_1.S_1$ is defined over the attribute Tp^p, and with an entity set of all the entities with $Sid \neq 2$. For ϕ_2, both the child and parent stripes are defined over the maximal entity set of the Sensor1 relation (i.e. for **all** tuples in Sensor1). Finally, for ϕ_3, the child stripe is defined over the attribute Tp^p of all the entities with $Sid = 2$, and the parent stripe is defined over Tp^p of all the entities with $Sid = 1$. In all cases, "1-1" mapping functions are assumed, which along with the corresponding CPTs, are omitted from the figure in the interest of space.*

Learning the shared correlation structure and the parameters of a FO probabilistic model from data is known to be a challenging task [19]. In an earlier workshop paper [14], we have discussed the complications of the learning process, and how it can be facilitated over a hierarchical version of FOBNs. Developing a complete, efficient learning solution for the BAYESSTORE FOBN model is in our immediate plans.

3 Query Processing

Having presented BAYESSTORE's probabilistic model, in this section we consider basic query processing algorithms from the standard SQL repertoire, namely selection, projection and join, extended with inference operations over the incomplete relations in a probabilistic database $\mathcal{DB}^p =< \mathcal{R}, F >$.

Unlike their relational counterparts, which operate only over deterministic relations, operators over a $\mathcal{DB}^p$have to process both the data in the incomplete relations and the possible-worlds distribution. A naive approach to query processing is to perform traditional relational operations on the set Ω of the exponentially many worlds, to compute a new set of possible worlds Ω', and correctly map probabilities from F to F'. Since this computation is clearly intractable, the core challenge is to develop query execution techniques which, prior to operating on the data of $\mathcal{DB}^p$'s incomplete relations, manipulate the first-order model accordingly. This preprocessing step guarantees that the resulting probability distribution F', encoded by the modified $\mathcal{M}_{FOBN}$, will be compatible with the new state of $\mathcal{DB}^p$.

3.1 Selection

We focus our discussion on the use of a selection predicate ϱ, that is a conjunction of atomic predicates ($\varrho = \wedge \varrho_i$), whose operators involve arithmetic comparisons $\Theta \in \{<, >, \leqslant, \geqslant, =\}$ between a probabilistic or deterministic attribute and a constant ($\varrho = A\Theta ct$). BAYESSTORE also supports other forms of atomic predicates, such as $\varrho = A^p \Theta B^p$ (presented in Section 3.3, where probabilistic joins are discussed), as well as boolean expressions involving disjunctions and negations, but details are omitted due to space restrictions.

The techniques for performing selection *over the model* (as we call the $\mathcal{M}_{FOBN}$ modification process) which will be discussed next, have proven to be quite fundamental for all the relational operations described in this section. On the other hand, selection over the tuples of R is not a trivial process either, as the filtering of the incompatible to ϱ tuples is no longer deterministic. Section 3.1.2 introduces a basic algorithm for selecting incomplete data tuples, as well as two optimizations, which attempt to reduce the size of the resulting incomplete relation R', without affecting the validity of the selection output.

3.1.1 Selection over Model $\mathcal{M}_{FOBN}$

In a deterministic DB, selection over a relation R reduces it to contain only the tuples that satisfy the selection predicate. Should R be an incomplete relation (e.g., the *Sensor1* relation of Figure 1, in which some temperature and light readings are missing), a selection with predicate ϱ : (*Sensor1*.L^p = `Brt`), apart from removing the tuples that have a light value different than `Brt`, affects the distribution of possible worlds F as well, since worlds where $L =$ `Drk` cannot be generated by ϱ.

In statistical model terms, this probabilistic selection operation resembles that of computing the conditional distribution $\mathsf{Pr}[\overrightarrow{X}|Y_1 = y_1, \ldots, Y_n = y_n]$ of a set of RVs $\overrightarrow{X}$, given that RVs $\{Y_i\}_1^n \in \overrightarrow{X}$, have specific values $\{y_1, \ldots, y_n\}$. Thus, it seems tempting to calculate the new possible-worlds distribution F' over the set of example RVs $\overrightarrow{X}_{Sensor} = \{Tp_1, \ldots, Tp_n, L_1, \ldots, L_n\}$, by conditioning on all the RVs that correspond to *Sensor1*.L = `Brt`. Unfortunately, this operation does not result in the correct possible worlds distribution: In a nutshell, the problem here is that conditioning (and other standard model manipulations, such as marginalization) cannot by themselves express possible worlds with different numbers of tuples — even under the conditioned/marginalized model, the number of tuples in each possible completion of an incomplete relation R is going to be the same (i.e., $|R|$). In contrast, applying the selection operation over the possible worlds *can obviously result in worlds with different numbers of tuples*, by filtering out tuples that do not satisfy the selection predicate; essentially, probabilistic selection introduces *tuple-existence uncertainty* (i.e., the *existence* of a tuple in a possible world becomes *uncertain*).

Continuing with our example, a given tuple $t \in$ *Sensor1* appears only in the possible worlds where the attribute $L =$ `Brt`, and in all the rest it is filtered out completely. On the other hand, by simply calculating the conditional $\mathsf{Pr}[\overrightarrow{X}_{Sensor}|L_1 =$ `Brt`$, \ldots, L_n =$ `Brt`$]$ directly over the model, the possible worlds corresponding to the resulting joint pdf all have *the same* number of tuples (i.e., the number of tuples in *Sensor1* with $L \neq$ `Drk`) all having $L =$ `Brt`. (It should also be intuitively clear that this (conjuctive) conditioning does not have the right semantics for our selection operation.) The fact that the *cardinality* of tuples between possible worlds can vary cannot be directly expressed by standard conditioning/marginalization operations on the model.

We capture tuple existence uncertainty by introducing the probabilistic attribute $Exist^p$ in the schema of the incomplete relation being selected, if it is not contained already. Depending on whether the selection predicate ϱ involves a deterministic A^d, or a probabilistic attribute A^p, the corresponding selection over a model $\sigma(\mathcal{M}_{FOBN})$ behaves differently.

The atomic predicate ϱ : $(A^d \Theta ct)$, restricts the entity set to which the predicate ϱ applies. For example, ϱ : $(Sid = 3 \wedge Tp =$ `Hot`$)$ restricts the entity set of predicate ϱ to only include the entities with Sid equals 3. Entities with $Sid \neq 3$ cannot exist in the resulting relation (i.e., their $Exist^p$=0).

Existence uncertainty is introduced in the model when ϱ involves a probabilistic attribute $\mathcal{A}^p$. For example, if a predicate $Tp =$ `Hot` is applied to a tuple t with a missing Tp value, the presence of t in the output becomes probabilistically dependent on $t.Tp$'s value. Intuitively, this operation corresponds to changing the local model of the factor that corresponds to R's $Exist^p$ attribute, to be dependent on that probabilistic attribute (for our example, Tp).

Figure 3 shows the `select-model` algorithm, based on a predicate ϱ of the form $A^d \Theta_1 ct_1 \wedge A^p \Theta_2 ct_2$. For example, $\varrho = (Sid = 3 \wedge Tp =$ `Hot`$)$. The algorithm can be easily extended to process a conjunctive predicate ϱ with any number of atomic predicates. (Join predicates (e.g., $A\Theta B$) are discussed in Section 3.3.)

Lines 1-2 define the entity set on which the selection predicate applies: $\varrho.\mathcal{E}$. The presence of the deterministic attribute inside ϱ ($Sid = 3$) essentially restricts its entity set $\varrho.\mathcal{E}$ (e.g., to all tuples with $Sid = 3$); otherwise, the predicate applies to any entity within

```
SELECT-MODEL (R, M_FOBN, ϱ = A^d Θ_1 ct_1 ∧ A^p Θ_2 ct_2)
1   M_FOBN' ← M_FOBN
2   ϱ.E ←< A^d Θ_1 ct_1 >
3   // Modification of pre-existing factors φ ∈ M_FOBN if φ.S_c
4   // over Exist^p – Step omitted due to space constraints.
5   if ϱ.E ≠ ∅ then
6      φ.S_c ←< R.Exist^p, ϱ.E >
7      φ.S_p ←< R.A^p, ϱ.E >
8      φ.f ← a one-to-one mapping between S_p and S_c
9      φ.CPT ← {Exist^p = 1 iff ϱ.A^p Θ_2 ct_2 == true}
10     M_FOBN' ← M_FOBN' ∪ φ
11  endif
12  if ϱ.E ≠ R.E then
13     φ.S_c ←< R.Exist^p, (R.E − ϱ.E ) >
14     φ.CPT ← {Exist^p = 0}
15     M_FOBN' ← M_FOBN' ∪ φ
16  endif
17  return M_FOBN'
```

Figure 3: Algorithm for selection over model with predicate $\varrho = (A^d\Theta_1 ct_1 \wedge A^p\Theta_2 ct_2)$

R's entity set.

If $\varrho.\mathcal{E}$ is non-empty, a new first-order factor ϕ is added to the model $\mathcal{M}_{FOBN}$ (lines 5-11). ϕ represents the local model of an $Exist^p$ attribute stripe over the entity set $\varrho.\mathcal{E}$: $S_{\ c} < Exist^p, \varrho.\mathcal{E} >$. $A^p\Theta_2 ct_2$ determines the parent stripe, $\phi.S_p$, of S_c. The latter is defined over the probabilistic attribute A^p in ϱ (e.g., a stripe on attribute Tp^p becomes a parent of a stripe on the $Exist^p$ attribute, with the same entity set, which in this example contains all the tuples with $Sid = 3$). A one-to-one mapping is established between $\phi.S_c$ and its parent stripe $\phi.S_p$, in line 8 (e.g., the value of the $Exist^p$ attribute in entity ε maps to the value of Tp^p in the same ε). Finally, the CPT table of ϕ is set to represent the deterministic conditional distribution: $Exist^p = 1$ with probability 1, iff $A^p\Theta_2 ct_2 = true$.

Lastly, lines 12-16 deal with the residual entity set (e.g., the entities with $Sid \neq 3$), which contains all the entities of R that are not included in $\varrho.\mathcal{E}$: ($R.\mathcal{E}$- $\varrho.\mathcal{E}$). A second first-order factor ϕ is defined over the $Exist^p$ attribute of the residual entity set, which specifies that none of these entities exists.

In the interest of space, the pseudocode of Figure 3 omits a preprocessing step that examines if there exist first-order factors in $\mathcal{M}_{FOBN}$ that are defined over attribute $Exist^p$. Such a case might arise if R is a result of a previous selection. For such an existence factor ϕ, the entity sets of its stripes are restricted to contain the entities that lie in the intersection of $\phi.S_c.\mathcal{E}$ and $\varrho.\mathcal{E}$. The correctness of the final possible-worlds distribution, follows trivially from the existence RVs added to the FOBN model.

EXAMPLE 4. *As an example, suppose the selection predicate of $\sigma_{Sid=3 \wedge Tp^p = \texttt{Hot}}(Sensor)$ is applied on the model $\mathcal{M}_{FOBN}$ in Figure 2. The selection operation first extends the schema of Sensor1 with a probabilistic attribute $Exist^p$, which represents the existence uncertainty of every tuple in Sensor1. It then modifies the model by adding two first-order factors over two different stripes of the $Exist^p$ attribute. The first specifies that for all entities with $Sid = 3$, $Exist^p$'s value is dependent on Tp^p's value: if $Tp^p = \texttt{Hot}$ then $Exist^p = 1$, and 0 otherwise. The second specifies that for all entities with $Sid \neq 3$, $Exist^p = 0$.*

3.1.2 Selection over Incomplete Relation $\sigma(R)$

In a traditional deterministic relation, a selection with predicate ϱ filters out exactly those tuples that do not satisfy the predicate. For an incomplete relation R, such a naive filtering operation would produce incorrect results. The first problem lies in whether to select the tuples with missing values in the predicate attribute: if the missing value has zero probability to satisfy the predicate, then the tuple should be filtered. A conservative approach would select all such tuples. The second problem is that there may be tuples in R that do not satisfy the predicate, but are still relevant, due to their correlation with another tuple that has non-zero probability of satisfying the predicate. The absence of these *evidence* tuples will lead to incorrect probabilistic inference computations later on.

As an example, consider two sensors, one placed on the exterior of a building, the other placed in a heating duct. The sensors are anti-correlated: when the exterior sensor is cold, the duct sensor is more likely to be hot. A query that asks for the temperatures of all sensors detecting heat, must retain the $Cold$ temperature reported by the exterior sensor, if the duct's sensor temperature reading in a given tuple is missing.

Thus, selection over an incomplete relation $\sigma(R)$ has to be *sufficient*, in retaining evidence tuples in R, which are relevant to any tuple that may satisfy the predicate with non-zero probability. On the other hand, selection should also be *minimal*, in deleting any tuple that neither satisfies the predicate nor is correlated with other "candidate" tuples. The "minimum" incomplete relation can be achieved by computing the marginal probability distribution over all the $Exist^p$ attribute values, and filtering all the tuples that have 0 probability of having $Exist^p = 1$. The complexity of this inference computation is known to be NP-hard, in general. We proceed to describe a base algorithm for $\sigma(R)$, and two optimizations that reduce the former's complexity, while being as economical in retaining evidence tuples as possible.

```
SELECT-DATA-BASE (R, M_FOBN, ϱ)
1   S ← SELECT(R, ϱ); R' ← ∅; E_1 ← S; i ← 1;
2   while E_i ≠ ∅ do
3      E_{i+1} ← ∅;
4      for each t ∈ E_i do
5         R' ← R' ∪ t; E_corr ← ∅;
6         // Locate correlated tuples to t by traversing M_FOBN
7         for each φ ∈ M_FOBN do
8            // If t is associated with an entity in one of φ's
9            // parent stripes...
10           if ∃j(j ∈ {1, ..., n}) : t ∈ φ.S_{p_j}.E then
11              E_corr ← E_corr ∪ f_j^{-1}(t);
12           // or child stripe...
13           else if t ∈ φ.S_c.E then
14              for each φ.f_j do
15                 E_corr ← E_corr ∪ f_j(t);
16        endfor endif endfor
17        E_{i+1} ← E_{i+1} ∪ E_corr;
18     endfor
19     i ← i + 1;
20  endwhile
21  return R'
```

Figure 4: Base Algorithm for select-over-data

Base Algorithm. Figure 4 displays the basic version of the data selection algorithm. According to the two objectives stated earlier, it selects tuples which either satisfy the predicate or contain missing values (tuple set T), while retaining all the tuples that are correlated with at least one tuple from T. The latter are determined by computing a transitive closure operation over T, using a *tuple correlation graph* – an undirected graph in which nodes correspond to R's tuples, and undirected edges represent correlations between pairs of tuples ("vertical" correlations).

In particular, the data selection algorithm initially uses the input

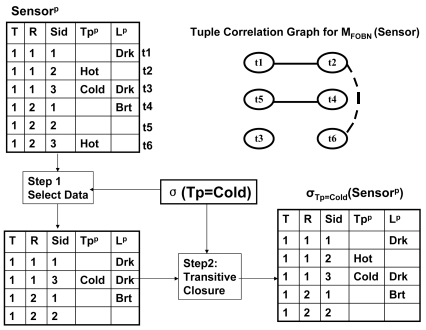


Figure 5: An example illustrating the Base Select-over-data Algorithm for $\sigma_{Tp='Cold'}(Sensor)$.

predicate ϱ to select the tuples in the incomplete relation R, which either contain evidence satisfying the predicate, or their probabilistic attribute value in the predicate is missing (line 1).

In lines 4-20, the transitive closure of the tuple set E_1 is computed, using semi-naive evaluation, over the tuple correlation graph. At the end of each iteration, a new set of tuples E_{i+1} is generated, containing the parents and children of each tuple $t \in E_i$, as defined in the correlation graph. At each iteration, the newly chased tuples are added to E_{corr}, the set of all correlated tuples with t.

Lines 7-16, represent the chasing of t's correlated tuples. All first-order factors ϕ in $\mathcal{M}_{FOBN}$ are traversed: if one of the *parent* stripes' entity sets $\phi.S_{p_j}.\mathcal{E}$ contains an entity that maps to t (or more formally, if the projection of t on $\phi.S_{p_j}.\mathcal{E}$'s schema belongs in $\phi.S_{p_j}.\mathcal{E}$), then the entity(ies) in the child stripe $\phi.S_c$ to which it maps, are computed through the reverse mapping $f_j^{-1}(t)$ and added to E_{corr}; if t can be associated with an entity of $\phi.S_c.\mathcal{E}$, then all its corresponding entities from each of $\phi.S_c$'s parents are chased as well.

EXAMPLE 5. *Figure 5 depicts the model $\mathcal{M}_{FOBN}$ of Sensor1, and the latter's tuple correlation graph. Tuples $\{t_1, t_2\}, \{t_4, t_5\}$ indicate correlated pairs. As a first step, the algorithm selects tuples from Sensor1 which satisfy the predicate or contain missing values. Consequently, it computes the incomplete relation Sensor1', calculating the transitive closure of the tuples generated from the first step, given the tuple correlation graph. Note that in the latter step, t_2 is added, because even though it does not satisfy the predicate, it has a vertical correlation with t_1.*

Evidence-Based Early-Stopping Optimization. The transitive closure computation that the base algorithm utilized to collect all vertically correlated tuples, despite its correctness, might conservatively select more tuples than necessary. However, two tuples t_1 and t_2 may be *conditionally independent*, even if there exists a vertical correlation between them. Consequently, we can accelerate the computation, by stopping early when encountering conditionally independent tuples, and thus reduce the number of resulting tuples.

Identifying conditionally independent tuples is done by means of the "Bayes Ball" algorithm [13]. Bayes Ball is a standard Graphical Model algorithm that takes as input a Bayesian Network and evidence values for some of its random variables, and determines pairs of nodes in the network that are independent, given the evidence. (The details can be found in the full version of the paper.)

EXAMPLE 6. *Continuing our example from Figure 5, let us suppose that another vertical correlation exists between tuples t_2 and t_6 ($t_2 \rightarrow t_6$), which is reflected with a dotted line in the tuple correlation graph. Although t_6 is correlated with t_2, the Bayes Ball algorithm indicates that it is conditionally independent with t_1, and therefore, the transitive closure stops at t_2; t_6 doesn't need to be chased.*

Factor-Based Data Filtering Optimization. Recall that the base algorithm conservatively maintained all the tuples that contained a missing value for the attribute mentioned in ϱ, assuming that the probability for such a tuple to have a value that satisfies ϱ is non-zero. Nevertheless, we could have used the $\mathcal{M}_{FOBN}$ model, conditioning on the evidence present in R, to verify if this holds for every such tuple. In particular, we have not yet utilized the probability distribution encoded in $\mathcal{M}_{FOBN}$, to actually verify if the probability of satisfying the predicate is non-zero, conditioning on the evidence present in the incomplete relation.

As an example, let us examine the following query over the *Sensor1* relation of Figure 5: $\sigma_{L=\texttt{Drk}(Sensor)}$. Suppose the local model of L, ϕ_2, has a conditional probability table ϕ_2.CPT, indicating that probability of $L = \texttt{Drk}$ is zero when $Tp = \texttt{Hot}$. Thus t_2, with its L value missing, has zero probability satisfying the predicate.

Since probabilistic inference is an expensive operation, as a relaxation, we avoid computing the conditional probability of the RV of *every* missing value, given all the available evidence in R and ϱ. Instead, we focus on tuples with RVs that are involved only in horizontal correlations. In particular, for each RV r corresponding to such tuples, we don't calculate the conditional over the whole model, $\mathsf{Pr}[r|\overrightarrow{X}_{ev} \cup \varrho.A^p = ct]$, where $\overrightarrow{X}_{ev}$ represents the vector of evidence data in R, but rather $\mathsf{Pr}[r|\overrightarrow{Y}_{ev} \cup \varrho.A^p = ct]$, where $\overrightarrow{Y}_{ev}$ is the vector of RVs from the same tuple as r, instantiated with the evidence in the tuple. This probability can be obtained by the CPT of the horizontal factor ϕ in which r belongs (as part of $\phi.S_c$). Should $\mathsf{Pr}[r|\overrightarrow{Y}_{ev} \cup \varrho.A^p = ct] = 0$, the tuple can be safely removed from the result set, and not chased for other correlated tuples to it.

3.2 Projection

In this section, we concentrate on projection $\pi_\Pi(R)$ without duplicate elimination, where the projection list $\Pi = \{A_1, .., A_n\}$ contains the primary key. This assumption guarantees that there will be no duplicates generated from the project operation, because all the primary key attributes remain in the resulting relation.

The inherent difficulty with duplicate elimination lies in the fact that it is inherently not a first-order operation. To be able to account for duplicates, we should model the uncertain quantities of the probabilistic DB *at the tuple level*, which would necessitate that we ground the first order model associated with the DB. We are actively exploring more concise probabilistic representations for the task of modeling duplicates, which is left as future work.

Traditional projection semantics over a deterministic relation, would result in retrieving each tuple from the relation, while keeping only a subset of its attributes. Such an operation over an incomplete relation though would generate incorrect results, because deleting the attributes that are not in the subset A may lead to loss of evidence, which is needed to generate the correct probabilistic distribution function F.

Not unlike selection, projection over an incomplete relation R retains some attributes A^* that are not in the projection list Π, if they are *correlated* with other attributes in Π, and deletes only the uncorrelated attributes $B = Sch(R) - \Pi - A^*$, where $Sch(R)$ is the schema (attribute list) of R. The project operation over the model $\mathcal{M}_{FOBN}$ deletes the first-order factors that involve attributes in the set B only.

The project algorithm is shown in Figure 6. In lines 1-5, the algorithm traverses all first-order factors ϕ in the model $\mathcal{M}_{FOBN}$.

PROJECT $(R, \mathcal{M}_{FOBN}, \Pi)$
1 $\mathcal{M}_{FOBN}' \leftarrow \mathcal{M}_{FOBN}$
2 **for** *each* $\phi \in \mathcal{M}_{FOBN}$ **do**
3 **if** $\exists i, j:\ \phi.S_i.A^p \in \Pi$ && $\phi.S_j.B^p \notin \Pi$ **then**
4 $\Pi \leftarrow \Pi \cup B^p$
5 **endif endfor**
6 **for** *each* $\phi \in \mathcal{M}_{FOBN}$ **do**
7 **if** $\forall i,\ \phi.S_i.A^p \notin \Pi$ **then**
8 $\mathcal{M}_{FOBN}' \leftarrow \mathcal{M}_{FOBN}' - \phi$;
9 **endif endfor**
10 $R' \leftarrow \emptyset$
11 **for** *each* $t \in R$ **do**
12 $R' \leftarrow R' \cup t(\Pi)$
13 **endfor**
14 **return** $R', \mathcal{M}_{FOBN}'$

Figure 6: Projection algorithm over Incomplete Relations

If ϕ involves two attributes A^p and B^p, where A^p belongs in the project list ($A^p \in \Pi$), and B^p does not ($B^p \notin \Pi$), then A^p and B^p are correlated by factor ϕ. Thus, on line 3, the algorithm retains B^p in project attribute set Π. In lines 6-9, the algorithm traverses all the first-order factors a second time, and processes the model $\mathcal{M}_{FOBN}$' to remove first-order factors that only involve attributes which are not in the newly computed set Π. In lines 11-13, the algorithm iterates through every tuple in the incomplete relation R, and performs a traditional projection of each tuple on the attributes that remained in Π.

EXAMPLE 7. *Using the example in Figure 5, suppose we have a new probabilistic attribute Humidity H, and a new first-order factor ϕ_4, which represents a prior probability table for all attribute values in H. Suppose we are to perform the projection $\pi_{\{T,R,Sid,L\}}$ on this modified incomplete relation* Sensor1*. Since the first-order factor ϕ_2 encodes a correlation between attributes Tp and L, Tp is then included in the projection attribute set Π. Attribute H is not included in Π, because it is not correlated with any attribute in Π. Thus, the model of* Sensor1 *is modified by deleting ϕ_4, because it only involves attribute H. Finally, the new incomplete relation is generated by projecting every tuple of* Sensor1 *on the subset of attributes $\Pi = \{T, R, Sid, Tp, L\}$.*

3.3 Join

We now turn our attention on binary selection predicates. In particular, we will examine how the join operator ($\bowtie_\varrho$) between two incomplete relations R_1 and R_2, whose entities' correlation patterns are captured by the first order model $\mathcal{M}_{FOBN}$, determines the contents of the resulting incomplete relation $R = R_1 \bowtie_\varrho R_2$, as well as the necessary modifications to $\mathcal{M}_{FOBN}$. We will concentrate on join predicates of the form: $\varrho = (R_1.A^p \Theta R_2.B^p)$, where attributes A and B are probabilistic.

3.3.1 Join over Model $\mathcal{M}_{FOBN}$

The modifications to $\mathcal{M}_{FOBN}$ which are required to maintain the consistency of the probability distribution F with respect to R, are quite similar to those that the selection over the model algorithm performs. As in section 3.1.1, we need to capture the *existence uncertainty* of every joinable pair of tuples $r_1 \in R_1$ and $r_2 \in R_2$, in the final result. The possible universe of these pairs corresponds to $R_1 \times R_2$. Nevertheless, these possibilities are restricted by the join predicate ϱ. Hence, the challenge, as in selection, lies in capturing the existence uncertainty of all *probable* tuple pair combinations in a *first order* fashion.

To accomplish this, we extend $\mathcal{M}_{FOBN}$ by adding a first-order factor over $R.Exist^p$, which depends on the probabilistic attributes

JOIN $(R_1(\overline{K}, \ldots, A^p), R_2(\overline{K'}, \ldots, B^p), \mathcal{M}_{FOBN}, \varrho)$
1 $R \leftarrow R_1 \bowtie_\varrho R_2$;
2 $\varrho.\mathcal{E} \leftarrow \pi_{\overline{K} \cup \overline{K}'}(R_1 \times R_2)$
3 *// Assign unique names to the stripes of the factors in $\mathcal{M}_{FOBN}$*
4 **for** *each* $\phi \in \mathcal{M}_{FOBN}$ **do**
5 $\phi.S_c \leftarrow$ RENAME$(\phi.S_c, Sch(R))$
6 **for** *each parent stripe* $\phi.S_{p_i},\ i = \{1, \ldots, n\}$ **do**
7 $\phi.S_{p_i} \leftarrow$ RENAME$(\phi.S_{p_i}, Sch(R))$
8 **endfor endfor**
9
10 *// Create an existence factor*
11 $\phi.S_c \leftarrow < R.Exist^p, \varrho.\mathcal{E} >$
12 $\phi.S_{p_1} \leftarrow < R_1.A^p, R_1.\mathcal{E} >$
13 $\phi.S_{p_2} \leftarrow < R_2.B^p, R_2.\mathcal{E} >$
14 $\phi.f_1 \leftarrow \forall \varepsilon_c \in \phi.S_c.\mathcal{E},\ \exists \varepsilon_1 \in \phi.S_{p_1}.\mathcal{E},\ \phi.f_1(\varepsilon_c) = \varepsilon_1 :\ \varepsilon_c.\overline{K} = \varepsilon_1.\overline{K}$
15 $\phi.f_2 \leftarrow \forall \varepsilon_c \in \phi.S_c.\mathcal{E},\ \exists \varepsilon_2 \in \phi.S_{p_2}.\mathcal{E},\ \phi.f_2(\varepsilon_c) = \varepsilon_2 :\ \varepsilon_c.\overline{K}' = \varepsilon_2.\overline{K}'$
16 $\phi.CPT \leftarrow \{Exist^p = 1\ \textit{iff}\ \varrho == true\}$
17 $\mathcal{M}'_{FOBN} \leftarrow \mathcal{M}_{FOBN} \cup \phi$
18 **return** $R, \mathcal{M}'_{FOBN}$

Figure 7: Join algorithm over Model $\mathcal{M}_{FOBN}$.

A^p and B^p that participate in ϱ. The algorithm is described in Figure 7. Initially, as a preprocessing step (lines 4–8), we rename the attributes in the schema of R, $Sch(R)$, to be able to refer to each attribute of R by a *unique* name. This renaming step needs to be carried over the attributes of the stripes of each first-order factor in $\mathcal{M}_{FOBN}$ as well, so that the correspondence of $\mathcal{M}_{FOBN}$'s stripes with the probabilistic attributes of R is maintained.

Lines 10–17 define the new existence first-order factor ϕ. Essentially, the entity set of ϕ's child stripe, $\phi.S_c.\mathcal{E}$ corresponds the cross product of the entities of the relations to be joined, $R_1(\overline{K}, \ldots, A^p)$ and $R_2(\overline{K}, \ldots, B^p)$, where $\overline{K}$ and $\overline{K}'$ are the sets of attributes comprising the primary keys of R_1 and R_2 respectively, as indicated by line 2. $\phi.S_c$'s parents are the stripes $\phi.S_{p_1}$ and $\phi.S_{p_2}$, representing the populations of RVs for the join attributes $R_1.A^p$ and $R_1.B^p$. Each entity ε_c from $\phi.S_c.\mathcal{E}$ is associated with the original "copies" of the two entities, ε_1 and ε_2, from which it was generated – one from each respective parent stripe. This many-to-one association between an entity of one of the two initial relations, and its multiple copies in the entity set of $\phi.S_c$, is captured by the mapping functions $\phi.f_1$ and $\phi.f_2$ (lines 14–15). Finally, the CPT of this new factor represents the deterministic conditional distribution $Exist^p = 1$ with probability 1, iff $R_1.A^p \Theta R_2.B^p = true$.

As a special case, if R_1 or R_2 are a result of a previous selection, in which case $\mathcal{M}_{FOBN}$ contains existence factor(s) already, a similar process takes place, which extends the entity sets of the child and parent stripes involved, as well as the condition defining the factors' CPTs, to the resulting cross-product domain of the join. A detailed description of this special case can be found in the full version of the paper.

3.3.2 Joining Incomplete Relations

As in the case of selection, the presence of missing values for the probabilistic attributes of an incomplete relation, causes complications when the latter is to participate in a join. In particular, when the attributes participating in a join predicate are probabilistic ($R_1.A^p \Theta R_2.B^p$), each tuple of R_1 (equivalently for R_2) that has a missing value for A^p, may satisfy ϱ and thus participate in the join result, depending on the possible values it can take, according to the $\mathcal{M}_{FOBN}$ and the evidence present in R_1 and R_2. Following the same intuition as in section 3.1.2, all such tuples from both relations should be considered "joinable" *conservatively*, as the join operation per se becomes uncertain.

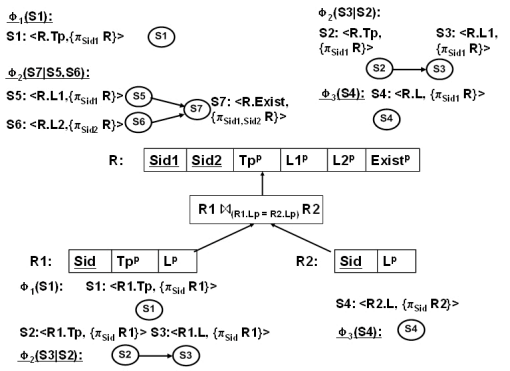


Figure 8: The join operation over the model of two incomplete relations, *Sensor*$_1$ and *Sensor*$_2$.

In addition, we might also need to include in the result the join of two tuples $r_1 \in R_1$ and $r_2 \in R_2$ that do not satisfy the join predicate, for the same reason this problem arises in selection; this resulting tuple might be correlated with another tuple from $R = R_1 \bowtie_\varrho R_2$ that satisfies ϱ, and hence its absence from R can erroneously interfere with probabilistic inference operators, should they later be run over it. Therefore, the set of these seemingly "disqualified" join tuples needs to be transitively calculated from the tuples of R' that satisfy ϱ, using the probabilistic information provided in $\mathcal{M}_{FOBN}$.

In essence, the above can be phrased as a direct extension of the selection over a single incomplete relation, studied in section 3.1.2. Taking advantage of the techniques developed there, as well as common Relational Algebra equivalences, we first compute the cross product of the relations to be joined, on top of which we apply our probabilistic selection operator, using the same predicate as in ϱ: $R_1 \bowtie_\varrho R_2 = \sigma_{\varrho=(R_1.A \Theta R_2.B)}(R_1 \times R_2)$.

EXAMPLE 8. *Given the example in Figure 8, the query we want to evaluate over these two incomplete relations is:*

```
SELECT *  FROM Sensor1 R1, Sensor2 R2
WHERE R1.L = R2.L
```

The algorithm initially renames the attributes in the definition of each factor's stripe S_i, so that they coincide with the resulting relation's schema: $R(\underline{Sid1}, \underline{Sid2}, Tp^p, L1^p, L2^p, Exist^p)$. A deterministic first-order factor ϕ_4 for $R.Exist^p$ is added to $\mathcal{M}_{FOBN}$. Its child stripe's entity set, $S7.\mathcal{E}$ is defined as: $\{\pi_{Sid1,Sid2}R\}$, while the parent stripes associated with it, $S5$ and $S6$, over the attributes $R.L1$ and $R.L2$ included in the join predicate, have entity sets: $\{\pi_{Sid1}R\}$ and $\{\pi_{Sid2}R\}$ respectively. Each entity in the parent stripe $S5.\mathcal{E}$ (likewise for $S6$) has a unique $Sid1$ value, $s1$, which is mapped with a set of children entities from $S.\mathcal{E}$, all having $Sid1 = s1$ and $Sid2 \in \mathsf{dom}(Sid2)$. Finally, the CPT of ϕ_4 is defined as:$(R.Exist^p = 1$, iff $R.L1 = R.L2)$.

3.4 Inference

We have studied two alternatives for incorporating probabilistic inference operators in BAYESSTORE. According to the first, rather naive alternative, one needs to *ground* the $\mathcal{M}_{FOBN}$ to a flat BN, on which standard inference algorithms are applicable [13]. A high-level description of our grounding algorithm is provided in Section 2.3. Although by taking this route we gain in algorithmic simplicity, we lose the advantage of the compact representation that a first-order model has to offer.

The second approach involves performing inference directly on the $\mathcal{M}_{FOBN}$. Its basis lies on an adaptation of a first-order inference algorithm, introduced by Poole in [16]. The first-order inference algorithm, operates on whole *populations* of random variables at once. Essentially, instead of multiplying the CPTs of a particular RV population individually, since all CPTs are the same, it raises their common CPT to the power of the population sizes; the existence of evidence tuples slightly complicates the above process. This technique avoids repetitive computation for random variables with the same local model, as in the case of inference over a standard BN. Our system already includes a small number of first-order optimizations, inspired by Poole's work. Nevertheless, a complete implementation of first-order inference inside BAYESSTORE is among our top priorities.

4 Experimental Evaluation

We have implemented a prototype version of BAYESSTORE that supports the selection variants, presented in section 3.1, and limited First-Order inference optimizations over FOBNs. The main findings of our experimental study can be summarized as follows:

- The aggressive data filtering performed by the proposed selection algorithms, reduces the sizes of the output relation (i.e., evidence tuples), thus enhancing inference efficiency.
- The utilization of First-Order inference techniques results in even further, *dramatic* acceleration of the inference stage over the FOBN.

Our implementation is based on Postgres 8.2.4 (`www.postgresql.org`) and uses Murphy's BNT Toolbox (`www.cs.ubc.ca/~murphyk/Bayes/bnt.html`) for inference. In this section we will describe the results of our experimental evaluation, which focused primarily on selection algorithms. All experiments were run on a 2.8Gz CPU, 1GB RAM system running the Fedora Linux Core 5.

4.1 Methodology

For our evaluation we are using an extension of the example scenario of Section 2 – a hotel DB with 10 rooms, each instrumented with 8 environmental sensors, measuring temperature and light level readings, which are all stored in the incomplete relation *Sensor*. The distribution F of this probabilistic database is captured by a $\mathcal{M}_{FOBN}$. In the interest of space, we chose not to show, but rather describe the model. Thus, $\mathcal{M}_{FOBN}$ contains 4 first-order factors: a prior for all temperature readings of the $1^{st}, 3^{rd}$ and 5^{th} – 8^{th} sensor for every room $\phi(Tp)$, one capturing the same correlation pattern between the light and temperature readings over every sensor in the hotel $\phi(L|Tp)$, and two first-order factors modeling the correlation of the temperature readings between the 1^{st} and 2^{nd}, and 3^{rd} and 4^{th} sensor pairs in every room, $\phi(Tp_2|Tp_1)$ and $\phi(Tp_4|Tp_3)$. The selection query $Q1$, that we use in all the experiments, was: '`SELECT * FROM Sensor WHERE L='Drk''`.

Data Generation. We generated a synthetic dataset, by forward sampling on the model described earlier. To assess our algorithms' efficiency, we varied a number of parameters related to properties of the $\mathcal{M}_{FOBN}$ used during query processing, and the relation *Sensor*. For the latter, we controlled its size '*size*' in number of evidence tuples generated, the ratio of missing values '*mratio*' among them, and the selectivity '*sel*' of Q over the data, by appropriately manipulating the CPTs of the first order factors $\phi(Tp_2|Tp_1)$ and $\phi(Tp_4|Tp_3)$. As far as the model is concerned, we experimented with its connectivity ratio '*cratio*' – the number of *vertical correlations* between tuples over the total number of tuples in *Sensor*. '*cratio*' was expressed by adding more first-order factors, on different stripes of *Sensor*, than the ones described in par. 4.1.

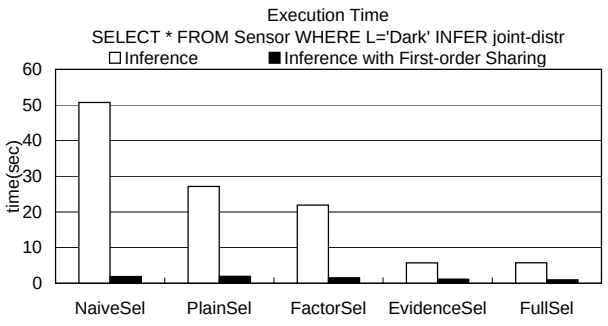


Figure 10: Inference execution time for query Q'. *size* =10000, *sel* =0.01, *cratio* =7/8, *mratio* =0.01.

Algorithms and Metrics. In this section, we present our experimental setting, which includes four selection algorithms over incomplete relations, incorporating the optimizations presented in Section 3.1.2. Algorithm `PlainSel` filters data tuples of an incomplete relation R using the traditional selection operator, and generates evidence tuples, resorting to full transitive closure over R and the model structure. `EvidenceSel` uses the "evidence-based early-stopping" technique on top of `PlainSel`, while `FactorSel` extends `PlainSel` by incorporating the "factor-based early-stopping" technique. Lastly, `FullSel`uses both optimizations.

We also use two baseline algorithms; `NaiveSel`, which keeps all tuples in the incomplete relation as evidence, and `DetSel`, which applies a traditional over R. As we argued in Section 3, `DetSel` will not produce correct answers. Nevertheless, we included it in our experimentation to compare its execution time (expected to be the smallest) against the proposed algorithms.

4.2 Scalability

In this first experiment, we varied the size of *Sensor*, while keeping the selectivity (0.01), the connectivity ($7/8$), and the '*mratio*' (0.01) constant. Because "evidence-based early-stopping" reduces the cost of the transitive closure computation and the size of the result relation, `EvidenceSel` and `FullSel` have shorter execution times (as in Fig. 9(a)) and smaller result sizes (as in Fig. 9(b)) than the `PlainSel` and `FactorSel` variants. By comparing `FactorSel` and `PlainSel`, we observe that the "factor-based data-filtering" technique employed by the former decreases the result size, but it increases the execution time. As expected, `DetSel`, has always the lowest cost, yet producing incorrect results.

4.3 Data Uncertainty

We also experimented by varying the ratio of missing values '*mratio*' in *Sensor*, while keeping its size to 10000 tuples, Q's selectivity to 0.01, and $\mathcal{M}_{FOBN}$'s connectivity ratio to 0.5. As indicated in Fig. 9(c), the execution time of the algorithms utilizing the "evidence-based early-stopping" technique grows more rapidly than the rest, as '*mratio*' increases. This behavior is justifiable, as a higher number of missing values indicates less evidence on which these algorithms can be based. Thus, on the higher end of the spectrum, `EvidenceSel` proves to be more expensive than the `NaiveSel`, and likewise `FullSel` is more expensive than `FactorSel`.

4.4 Model's Connectivity Ratio

The third parameter we varied was the model's connectivity ratio, keeping this time *Sensor*'s size (10000) and '*mratio*' (0.1), and Q's selectivity (0.1) constant. Figures 9(e) and 9(f) clearly demonstrate that the execution time and the result relation's size of `PlainSel` and `FactorSel` increase linearly with respect to connectivity, because they do not utilize the "evidence-based early-stopping" technique. On the other hand, '*cratio*' of the model has little effect on `EvidenceSel` and `FullSel`, with a minimal increase in the result size of around 10%.

4.5 Inference

One of BAYESSTORE's major objectives is the integration of relational queries with probabilistic inference operators. In our final experiment, we make use of an SQL extension which computes the joint distribution F of the RVs that correspond to the missing values of *Sensor*, for all the readings that satisfy the condition `L='Drk'`. A presentation of our proposed SQL extension can be found in the extended version of our paper. Thus, we are executing the query Q': '`SELECT * FROM Sensor WHERE L='Dark' INFER joint-distr`', which according to our query semantics, will produce a new incomplete relation *Sensor*' and model $\mathcal{M}_{FOBN}$'.

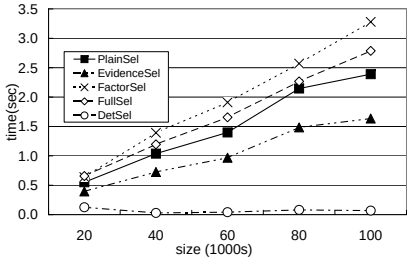
The selection of Q' operates both on the model $\mathcal{M}_{FOBN}$, using the `Select-Model` algorithm of paragraph 3.1.1, and on *Sensor*, using the different data selection algorithms of our experimental framework. The resulting model $\mathcal{M}_{FOBN}$' is grounded to a flat Bayesian Network, so that standard inference algorithms can be applied. Due to time limitations, rather than integrating an inference operator in our prototype, we used the implementation of the Variable Elimination algorithm from Kevin Murphy's BNT toolbox. On top of it, as we briefly describe in Section 3.4, we implemented one of the First-Order inference techniques discussed in [16] (`FirstorderSharing` technique), which allow us to share the inference computation over a population of instances with the same model template, without fully grounding the model. Suppose a sensor deployment, where every sensor has an independent and identical probabilistic model. The inference computation is identical, and thus can be shared among all sensor readings with the same probabilistic attribute values.

Fig. 10 shows the total execution time of Q', using our various data selection algorithms, with and without our first-order optimizations. The latter result to reduction on the query execution time. The execution time decreases as we apply more complicated optimizations in the selection algorithm. With `FirstorderSharing`, `NaiveSel`, which keeps the full incomplete relation as evidence, takes the most processing time. `FullSel` completes in only half the time, compared to `NaiveSel`. The decrease is more dramatic without first-order optimization. These results indicate that data and evidence filtering techniques manage to reduce the time of the inference operator, as a welcome effect of the incomplete relation's size reduction. The increased cost for the select operation is almost ten times less than the reduced cost for the inference operation.

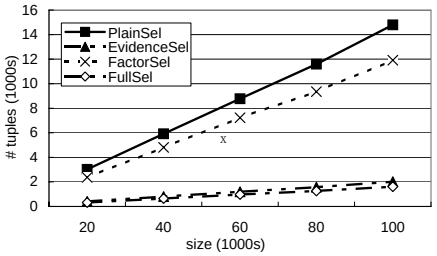
5 Conclusions and Future Work

In this paper we presented BAYESSTORE, a Probabilistic Database System that draws results from the Statistical Learning literature, to efficiently express and reason about correlations among uncertain data items, in a concise and statistically sound way. BAYESSTORE is based on a novel data model, comprised of a set of *Incomplete Relations* and a probability distribution over those relations. Thus, our PDBS manages to fully expose both the incomplete relations *and their distribution* to the user, as a unified, robust probabilistic representation that can be jointly manipulated. Furthermore, it represents one of the first successful attempts to seamlessly integrate state of the art Statistical Machine Learning techniques with relational query processing.

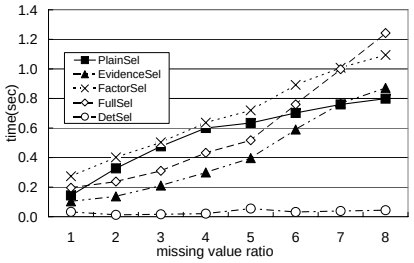
To represent probability distributions compactly, we use *First-Order Bayesian Networks* (FOBNs), which use a small set of first-



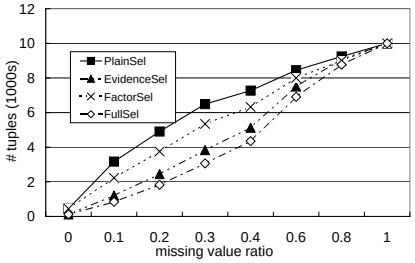
(a) Execution time, varying R's size, while keeping $\{sel = 0.01, cratio = 7/8, mratio = 0.01\}$ constant.



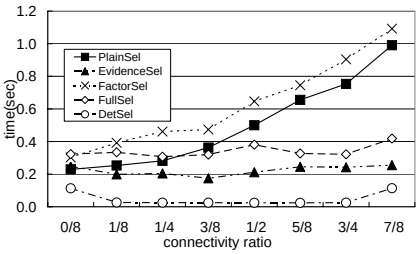
(b) Result size, varying R's size, while keeping $\{sel = 0.01, cratio = 7/8, mratio = 0.01\}$ constant.



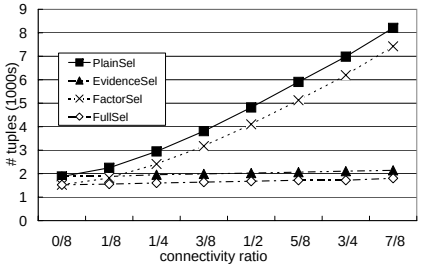
(c) Execution time, varying the missing values' ratio, while keeping $\{size = 10000, sel = 0.01, cratio = 1/2\}$ constant.



(d) Result size, varying the missing values' ratio, while keeping $\{size = 10000, sel = 0.01, cratio = 1/2\}$ constant.



(e) Execution time, varying the model's connectivity ratio, while keeping $\{size = 10000, sel = 0.1, mratio = 0.1\}$ constant.



(f) Result size, varying the model's connectivity ratio, while keeping $\{size = 10000, sel = 0.1, mratio = 0.1\}$ constant.

Figure 9: Experimental results for varying the incomplete relation's size (a-b), the predicate's selectivity (c-d), and the model's connectivity ratio (e-f).

order factors to capture correlation patterns between entire *populations* of uncertain entities. Furthermore, we have illustrated both algorithmically and experimentally, how FOBNs can be used to expedite query processing in large-scale datasets, by establishing *probabilistic inference* as a first class citizen in the operators repertoire that a PDBS needs to support. We present a significant first step toward integrating traditional query optimization techniques with inference, by showing how to do aggressive early selection on both evidence and probability models. Simple experiments illustrate that this can easily lead to an order of magnitude performance improvement over schemes that do not reason about the probability model, and the conditional independence of data items.

Our query processing results are suggestive, and more remains to be done. As we discuss, an open challenge is to come up with efficient algorithms for first-order, duplicate-free projections, that do not need to work on the tuple level. In addition, our evaluation motivates the implementation of efficient first-order inference algorithms as a new class of *probabilistic* query operators, that can be tightly integrated with the rest of the query processing engine, so that they can be made *optimizer-aware*, and reuse some of the existing functionality, provided by traditional relational operators.

6 References

[1] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, 2008.

[2] L. Antova, D. Olteanu, and S. Scherzinger. $10^{(10^6)}$ worlds and beyond: Efficient representation and processing of incomplete information. In *ICDE*, 2007.

[3] D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE TKDE*, 4(5), 1992.

[4] O. Benjelloun, A.D. Sarma, A. Halevy, and J. Widom. ULDB: Databases with Uncertainty and Lineage. In *VLDB*, 2006.

[5] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *VLDB*, 1987.

[6] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[7] N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*, 2004.

[8] A. Deshpande and S. Madden. MauveDB: Supporting Model-based User Views in Database Systems. In *SIGMOD*, 2006.

[9] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In *IJCAI*, 1999.

[10] N. Fuhr and T. Rolleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. In *ACM TOIS*, 15(1), 1997.

[11] R. Gupta and S. Sarawagi. Curating probabilistic databases from information extraction models. In *VLDB*, 2006

[12] T. Imieliński and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4), 1984.

[13] M.I. Jordan. Graphical models. *Statistical Science (Special Issue on Bayesian Statistics)*, 19:140–155, 2004.

[14] E. Michelakis, D.Z. Wang, M. Garofalakis, and J.M. Hellerstein. Granularity conscious modeling for probabilistic databases. In *DUNE*, 2007.

[15] A. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford, 2000.

[16] D. Poole. First-order Probabilistic Inference. In *IJCAI* , 2003.

[17] P. Sen and A. Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In *ICDE*, 2007.

[18] P. Sen, A. Deshpande, and L. Getoor. Representing tuple and attribute uncertainty in probabilistic databases. In *DUNE*, 2007.

[19] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, 2002.