

Chapter 2

OPTIMAL CONSTRAINT MANAGEMENT IN DISTRIBUTED DATABASES

2.1 Introduction

2.1.1 Local Verification of Global Integrity Constraints

Increasingly, enterprise-wide information systems are being built in distributed, heterogeneous environments. The prevalence of the Internet and the World Wide Web [BLCea94] allow designers to incorporate data and information from multiple sources. In those systems centralized control may be difficult, if not impossible, due to the autonomy of the local constituents, which indicates that highly autonomous federated distributed architectures [KGea96] are more appropriate. Often, workflow coordination for distributed systems involves constraint-based agreements, which can be viewed as global integrity constraints [JK97]. These global database integrity constraints are difficult to monitor, update and enforce in distributed environments, so that new,

distributed techniques and protocols are desirable.

To reduce the costs of distributed management of global constraints, the idea of local verification of global constraints was introduced and studied (e.g., [BGM92, GM91, GW93, Maz93, Qia89, SV86, GSE⁺97]). The idea is to decompose a global constraint into a set of local ones that will serve as a conservative approximation, that is, satisfaction of local constraints by a database instance guarantees satisfaction of the global constraint. Then, when a local site i is being updated, if the update satisfies its local constraint C_i , no global constraint checking is necessary. Thus, most of the work can be delegated to local processing, thereby saving communication and other distributed processing costs. The ability to perform updates autonomously is also very important in presence of site or network failures.

While the above-mentioned works have considered many aspects of local verification (see Related Work section), they have not addressed the problem of finding optimal constraint decompositions and distributed constraint-management protocols that achieve decomposition optimality along with maximal resource utilization. This is precisely the subject of this chapter.

To illustrate the problem of distributed constraint management we now consider an example of a distributed database for an application of logistics support for crisis management.

2.1.2 Crisis Management Scenario Example

Emergency service providers (e.g., fire fighters, medical personnel, military etc.) must be prepared to respond efficiently to crises such as floods, fires and earthquakes. In order to perform Crisis Management, the enterprise must find, coordinate, allocate, deploy and distribute various resources (such as food, clothing, equipment, emergency personnel, transportation) to the victims of a crisis. These resources are typically geographically distributed among many warehouses, suppliers, military units, local fire departments, bus terminals, etc. Each location may maintain a *local database* that stores and monitor information about available resources and their quantities. Our *distributed database* in this example is a loosely-connected collection of local databases, which are related, however, because of a global constraint on resources.

The global constraint in such a distributed database may originate, for example, from a number of pre-defined crisis management scenarios that require that certain amounts of resources be delivered to any potential disaster area within bounded time using available transportation. For example, a Hurricane Relief Mission to Florida may require that the following resources be delivered there in 24 hours: 1) sufficient canned food to feed 30,000 people for 4 days, 2) a supply of tents to support a tent city of 20,000 people, 3) medicines and vaccines to inoculate the tent city residents against cholera, 4) computers and communications equipment to support the coordination, command and control functions of the mission, 5) 10 medical units with medical personal and portable facilities to care for victims, and 6) DoD personnel to staff the mission.

For this scenario, the global integrity constraint would reflect that, for each resource type above, the overall amount of this resource available in all locations reachable in 24 hours (with the available transportation) is greater than or equal to the amount required in the scenario. Note that some resources are composed of other resources, which also needs to be reflected by the global constraint. For example, each of the required 10 medical units (resource 1) is composed of 2 MD's (resource 2), 5 paramedics (resources 3) and must have 2 tents (resource 4), 2000 vaccination packages (resource 5), 500 first aid packages (resource 6), etc. In turn, each vaccination package may be composed of certain quantities of other items and so on.

When a local site of the distributed database is being updated, for example when a certain amount of materiel is taken from a warehouse (not necessarily for crisis support), the update can only be allowed if it satisfies the global integrity constraint, which depends, in general, on the global database instance, not just on the updated local instance. Therefore, verification of the global constraint would require a distributed transaction involving possibly hundreds of loosely connected distributed sites, which might be an extremely expensive and time-consuming operation, especially when protocols such as two phase-commit are used to guarantee the standard properties of transaction atomicity, consistency, isolation and durability (e.g., [JK97]). Moreover, such distributed transaction would often not be possible in the presence of site and network failures, whereas the robustness feature, i.e., the ability to operate in the presence of (partial) failures, is crucial for applications such as Crisis Management. In

short, protocols managing local verification of the global constraint can significantly reduce distributed processing costs and increase the system robustness in the presence of failures.

2.1.3 Contributions

This chapter focuses on the problem of deriving the best possible decompositions, during both database design and update processing. It formulates a generic and powerful framework for finding optimal decompositions for a range of design and update-time scenarios, and provides a comprehensive solution for the case of general linear constraints, which are widely used in distributed applications such as resource allocation, reservations, financial transactions, and logistics. The comprehensive optimization-based solution includes (1) reducing the problem to mathematical programming, (2) developing algorithms for it, and (3) providing a distributed protocol to manage local updates and concurrent distributed constraint decompositions in the presence of communication and site failures, while guaranteeing the desirable properties of consistency, safety, optimality and last-resort update refusal.

More specifically, the contributions of this chapter are as follows. First, we introduce a generic optimization framework to achieve best decompositions by defining (1) the solution space of all *feasible* decompositions (explained below) (C_1, \dots, C_M) of the global constraint Ω over M distributed sites, and (2) the objective function that can describe a variety of optimization criteria, such as the probability that an update

satisfies its local constraint, the expected number of updates before the first update that violates a local constraint, or the expected overall cost of operations during an update.

The solution space of all *feasible* decompositions is the set of decompositions having the first and possibly other properties from the following list (depending on what is known at the time of a decomposition):

1. *Safety*, i.e., satisfaction of local constraints by a database instance must guarantee satisfaction of the global integrity constraint.
2. *Local Consistency* w.r.t. to a given database instance, i.e., each local instance must satisfy its local constraint (i.e., at the same local site). Clearly, local consistency and safety imply *global consistency*.
3. *Partial-constraint preservation* w.r.t. a given subset θ of sites and local constraints for sites outside θ , i.e., the decompositions cannot change the given local constraints outside θ .
4. *Resource partition* B_θ w.r.t. to a subset θ of sites. This property is based on the notion of *resources* and their *upper bounds* (explained below) associated with each local and the global constraint. Resource partition means that the global constraint resource upper bound is partitioned between the sites in and outside θ , and the cumulative *resources* of sites in and outside θ must be bounded by their corresponding upper bounds. The notion of resource partition is more flexible

than constraint preservation, and allows concurrent constraint (re-) decompositions.

One or more properties 1-4 are required for various decomposition scenarios, depending on what is known at the time of a decomposition. For example, to design a Crisis Management Database schema and local constraints when no actual database instance is known, the property of *safety* is required, while *local consistency* is not applicable. Often, only a partial design is required when local constraints for most sites have already been fixed, in which case the property of *partial constraint preservation* is needed in addition to *safety*. Assume now that the Crisis Management Database is operational, and the current local constraints entail the global constraint (i.e., *safety*) and the current database instance satisfies the local constraints (i.e., *local consistency*). Consider an update at site i , for example when a certain number of blankets is being taken from a warehouse, and the number of remaining blankets, stored at local database site i , has to be updated. If the update satisfies the current local constraint at site i , no processing except for the update itself is necessary, because *safety* guarantees that the global constraint is satisfied. However, if the update violates the local constraint (i.e., *local consistency*) *no longer holds*, a protocol can try to find a new feasible decomposition of the global constraint that will regain *local consistency* and still be *safe*. A more sophisticated protocol may try to re-decompose constraints only in a (hopefully small) subset θ of (well-connected) sites, which will be done under the assumption that the cumulative resources in and outside θ will stay within their corresponding resource

upper bounds, i.e., a new feasible decomposition will have the property of *resource partition*, in addition to *safety* and *local consistency*.

Second, for the case of general linear arithmetic constraints, we reduce the optimization-based framework to a standard, finitely-specified problem of mathematical programming. This is done by proving existence and actually developing a finite parametric (i.e., in terms of coefficients) characterizations of the properties 1-4 of feasible decompositions together with optimization criteria ¹, as follows:

- *Compact Split Decompositions*. Given a global constraint Ω , a parametric characterization of safe decompositions mean formulating a constraint $D(\vec{w})$ whose variables \vec{w} are the parameters (i.e., coefficients) of local constraints, such that $D(\vec{w})$ is true precisely for all safe decompositions. The problem, however, is that in general a constraint C_i at site i may be characterized by an unlimited number of atomic linear constraints; thus the size of a parametric description (using coefficients of those constraints) is unbounded. To overcome this problem, we introduce the notion of *compact split* safe decompositions, for which we prove that: (1) there does exist a parametric description of bounded size and (2) the optimum of any monotonic function ² among all safe decompositions can always be found in the subspace of compact split safe decompositions.

¹Not every family of constraints have such finite characterization, but we prove that the linear constraints do.

²We claim that any reasonable "decomposition quality" objective function must be monotonic, i.e., intuitively, the more databases instances a decomposition satisfies, the better.

- *Reducing Decompositions to Resource Distributions.* We introduce a *resource-based* characterization of split decompositions to reduce the problem of decomposing constraints to the problem of distributing resources, which significantly simplifies the distributed management of constraints. Specifically, every local constraint C_i for site i in a split decomposition D is uniquely associated with a *resource* vector³ r_i , and the global constraint is associated with the *global resource* vector, for which we prove that: D is a compact split (safe decomposition) if and only if the cumulative *resource* in all sites is bounded by the *global resource*. Furthermore, given a database instance, every site i is also associated with a *lower resource bound*, for which we prove that: the local database instance at i satisfies its local constraint if and only if its *resource* is bounded from below by its *lower resource bound*. In addition, every site is associated with its *resource upper bound*. The *resource* and its *bounds* for every site constitute a *resource distribution*, which a protocol can maintain instead of explicit local constraints and database instance. The key advantage of a resource distribution is its small size of $O(nc)$ as compared with the size $O(nc*nv)$ of a constraint decomposition, where nc and nv are the number of constraints and variables, respectively, in the global constraint. In fact, nv may be as large as the size of a database, for example when the global constraint reflects that the summation of some quantity, one per relational tuple, is bounded by a constant.

³the dimension of this and other resource vectors equals to the number of atomic linear constraints (i.e., linear inequalities over reals) in the global constraint.

- *Concurrent Split Decompositions.* To manage concurrent constraint decompositions, a protocol needs to be able to (re-)decompose constraints autonomously in a (small) subset θ of sites, when the constraints and database instances outside θ are unknown, and, furthermore, may change.⁴ The only imposed limitation is the property of *resource partition* B_θ w.r.t. θ , that is, the cumulative resources of sites in and outside θ must be bounded by their current *resource upper bounds* B_θ and its complement, respectively. We show that the decompositions can be done autonomously in θ by proving that, given a database instance for sites in θ , the following are equivalent: (1) there exists a (partial) *permissible* resource distribution for sites in θ w.r.t. B_θ , i.e., such that for each site in θ its *resource* is bounded between its *lower* and *upper bounds*, and that the cumulative *resource upper bound* in θ is exactly B_θ , and (2) there exists a (full) compact split (safe decomposition) of the global constraint that satisfies *resource partition* w.r.t. B_θ and local consistency. Furthermore, we show that optimal constraint decomposition adhering to *resource partition* can also be achieved autonomously in θ . Specifically, we prove that, given any (1) *resource partition* B_θ , (2) local constraints outside θ (and possibly (3) a database instance satisfying the local constraint outside θ), an optimal (partial) safe decomposition in θ adhering to the *resource partition* B_θ and, possibly, to *local consistency* w.r.t. the database instance yields a (full) safe decomposition that is optimal among all safe decom-

⁴Note that constraint preservation property is not adequate for this purpose because it assumes that the constraints outside θ are fixed (and known).

positions that hold the same properties plus the *partial constraint preservation* w.r.t. the local constraints outside θ . Moreover, we show that combining optimal (partial) safe decompositions for sites in and outside θ that satisfy *resource partition* B_θ and, possibly, *local consistency* yields an optimal (full) safe decomposition with the same properties.

- *Parametric Characterization of Objective Function and its localization.* While the parametric characterization of compact split decompositions is applicable to any monotonic objective function, we consider a specific optimization function in more detail: *maximizing the probability of not violating local constraints*. Specifically, we provide an analytical expression of this probability function in terms of parametric characterizations of compact split decompositions (i.e., resources), which we do by using the polyhedron volume function [CH79, Las83, Bea96], under the uniform distribution assumption of database instances, described precisely in Section 2.6. We also express this function in terms of partial resource distribution, so that the optimization could be done within a (small) subset of sites.

Third, we actually develop and partly implement an algorithmic framework to solve the resulting mathematical programming problems, for the case of maximizing the probability of not violating local constraints. For this case, the constructed optimization problems in terms of parametric descriptions have linear constraints and a non-

linear objective function, which is based on a parametric representation of the volume function. For safe decompositions, when each local constraint is in a single variable, the constructed objective function turns out to be concave; this property enables us to use a global search algorithm. We adopt the Frank-Wolfe algorithm [BS79, Kam84] to solve it. For other cases, the objective function is not concave and we use local search techniques in the algorithmic framework, that incorporate the Frank-Wolfe algorithm for search in local neighborhoods. To run experiments and prove the feasibility of the approach we have implemented an optimization engine for safe decompositions with local constraints in single variables. The experiments suggest that the approach is feasible and scalable, but more experimental study will be necessary to fine-tune and extend the algorithms for various specific cases.

Fourth, in order to exemplify the use of constraint decomposition techniques, we develop a distributed (tunable) protocol to manage resource distributions (i.e., local updates and concurrent distributed constraint decompositions) in the presence of site and network failures. We formulate desirable properties to hold for such a protocol, namely, local and global *Consistency*, decomposition *Safety* and *Optimality* and *Last-resort update refusal* (CSOL-properties), and come up with Distributed Protocol Assumptions under which, we prove, the CSOL-properties must hold. The suggested protocol satisfies the assumptions and thus possesses the CSOL-properties, but many other protocols are possible. In particular, our results are readily available to extend, with a significantly more powerful class of constraints and the guaranteed CSOL prop-

erties, a variety of database protocols, including [BGM92] for distributed databases and [SS90] for supporting local transactions in the presence of network partitions. Also, our results on decomposing constraints can easily extend [SK95, SK97] dealing with quasi-views, where global conditions (constraints) for re-materialization can be decomposed among subviews.

2.1.4 Related Work

The body of work on constraints in databases is too large to attempt to survey. The problem of integrity constraint verification has drawn much attention (e.g., [BGM92, GM91, GW93, Maz93, Qia89, QS87, SV86]). This includes local verification of global constraints in distributed databases (e.g., [BGM92, GM91, GW93, Maz93, Qia89, SV86, GSE⁺97, Huy97]). However, none of the works on local verification, to the best of our knowledge, has solved the problem of optimal selection of local constraints.

More closely associated with our work are the works [BGM92, SS90, MY98] which deal with numerical constraints, and the works [Las, LM92, HJLL] which consider parametric linear constraint queries and their connection to Fourier’s elimination method; the work on parametric queries, however, assumes that the number of parameters (i.e., coefficients) is bounded, which is not the case for our *safe* decompositions.

Perhaps most closely related is the work of [BGM92], which was the first to consider verification of linear arithmetic constraints in the context of distributed databases. It considers a single atomic linear constraint at the global level. Intuitively, an atomic

constraint must be such that it could be decomposed between two sites storing individual variables using some constant *boundaries*⁵. For example, the global constraint $A + B \geq 100$ can be decomposed into two local constraints $A \geq a$ and $B \geq b$, where a and b are constants such that $a + b \geq 100$; or the global constraint $A \leq B$ can be decomposed into $A \leq a$ and $b \leq B$, where $a \leq b$. The focus in [BGM92] is on the “demarcation” distributed protocol which is concerned with efficient (in terms of communication and other costs) negotiation between two sites on synchronizing the change in constant boundaries, in case a local update violated its local constraint.

The work [SS90] uses an idea similar to the demarcation of [BGM92]⁶, in the context of network partition failures, in order to overcome the problem by trying to perform transactions locally. Similar to [BGM92], a global constraint in the example considered in [SS90] is a single linear inequality of the form $x_1 + \dots + x_n \leq c$, which is split among n sites (i.e., a single variable per site) by giving each site a quota of c . However, [SS90] focuses on the distributed transaction management and leaves the problem of how to achieve constraint decompositions, as well as the question on what constraint families its techniques are applicable open.

The recent work [MY98] extends the demarcation protocol of [BGM92] by considering a wider class constraints: linear, quadratic and polynomial constraints. A global constraint is a single inequality that is decomposed into local constraints involving one variable per site. Since for this case, the property of *safety* corresponds geometrically to

⁵There is no precise formulation of the allowed atomic constraints in [BGM92].

⁶In fact, [SS90] is earlier.

containment of a multidimensional rectangle in the shape described by the global constraint (inequality), [MY98] suggests the use of geometrical techniques for (dynamic) decompositions. However, geometrical techniques (e.g., from computational geometry) are restricted to low dimension (i.e., small overall number of variables) whereas typically distributed databases involve a large number of variables used in the global constraint (e.g., Crisis Management Scenario).

In contrast to [BGM92, SS90, MY98], our methods on linear arithmetic constraints have none of the above-mentioned restrictions, i.e., we allow atomic linear inequalities of any general form, a global constraint may have any number of atomic constraints, constraints may be partitioned among any number of sites, and each site may have not just one, but any number of variables. Furthermore, ours is the only work suggesting achieving optimal decompositions, and providing a comprehensive solution for it. Moreover, our decompositions can work for different scenarios, i.e., with different assumptions regarding what is known at the time of a decomposition.

The work [Maz93] dealt with first-order (not numerical) constraints, in the context of distributed databases, and suggested certain heuristics to select better decompositions. However, the questions of how, and under what conditions, these heuristics relate to optimization criteria such as maximizing the probability of not violating local constraints and the optimality of decomposition was not considered.

The work [QS87] has also considered a certain class of first-order (not numerical) constraints in the context of equivalent reformulation of a constraint (which is dif-

ferent from our *safety*) in the presence of additional semantic information (not for a distributed environment). They also suggested some heuristics, based on costs of constraint verification and reformulation, but no algorithm or guarantee of optimality in any sense was provided. Finally, [Qia89] applied the techniques of [QS87] for equivalent constraint reformulation in the context of distributed databases.

2.1.5 Organization

The chapter is organized as follows. Following the introduction, Section 2.2 provides a formal framework for selecting optimal decompositions, which is generic for all types of constraints. In Section 2.3 we then concentrate on linear arithmetic constraints. Section 2.4 concentrates on parametric characterizations for the case when each distributed site has just one variable, while Section 2.5 considers the case of unrestricted variable partitions. Section 2.6 discusses the local *uniformity* assumptions on the update space, a specific optimization function and its localization. In Section 2.7 we describe the distributed protocol manage global (integrity) constraints and its properties. Section 2.8 focuses on actual algorithms, implementation and experiments with a number of decomposition examples.