Multidatabase Interdependencies in Industry

Amit P. Sheth
Bellcore†
444 Hoes Lane
Piscataway
NJ 08854-4182
amit@ctt.bellcore.com

George Karabatis*
Department of Computer Science
University of Houston
4800 Calhoun Rd
Houston, TX 77204-3475
george@cs.uh.edu

Abstract

In this paper we address the problem of data consistency between interrelated data. In industrial environments, lack of consistent data creates difficulties in interoperation between systems and often requires manual interventions to restart operations that fail due to inconsistent data. We report the results of a study to understand applicability, adequacy and advantages of a framework we had proposed earlier to specify interdatabase dependencies in multidatabase environments. We studied several existing Bellcore systems and identified examples of interdependent data. The examples demonstrate that the framework allows precise and detailed specification of complex interdependencies that lead to efficient strategies to enforce the consistency requirements among the corporate data managed in multiple databases. We believe that our specification framework can help in the maintenance of data that meet a business's consistency needs, reduce time consuming and costly manual operations, and provide data of better quality to end users.

1 Introduction

Poor specification and maintenance of consistency between related data may hamper interoperability and result in wasted operations, leading to manual intervention or system patches that incur higher expense. Consistency of corporate data requires managing interdependent data which are logically related data maintained by multiple databases. Previously, we proposed a framework for specifying interdependent data [RSK91][SR90]. In this framework, the interdependency between multiple systems is represented using Data Dependency Descriptors (D^3). A D^3 is a 5-tuple:

$$D^3 =< \mathcal{S}, U, P, C, \mathcal{A} > \text{ where:}$$

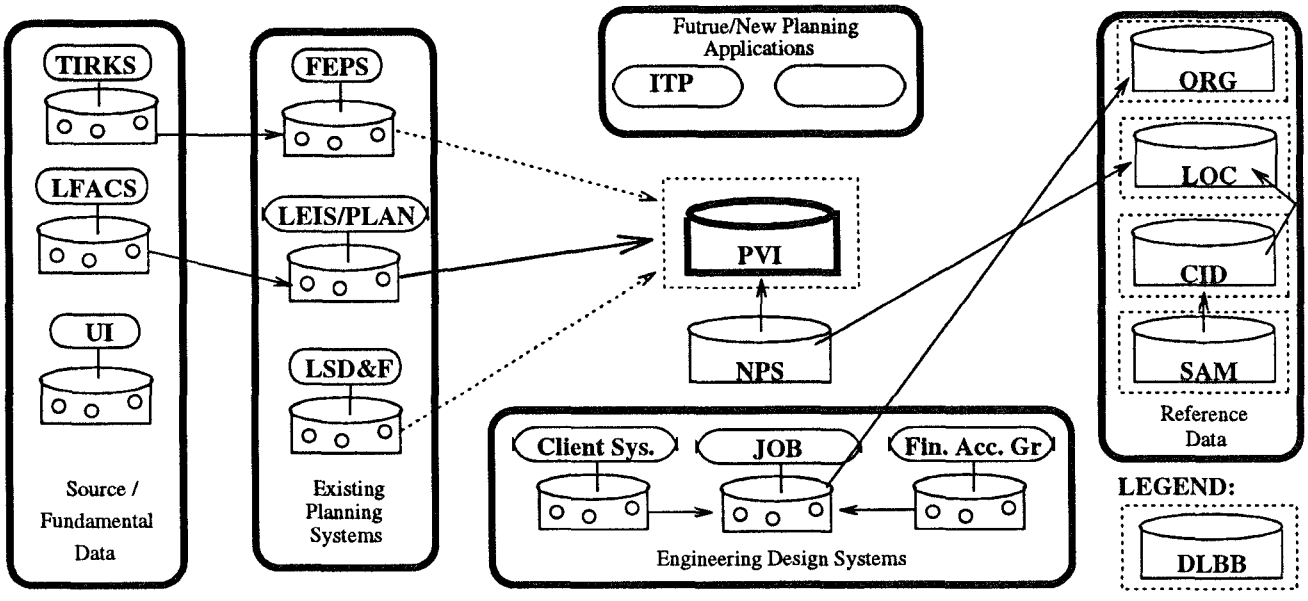
- $\mathcal{S}$ is the set of *source data objects*,
- U is the *target data object*,
- P is a boolean-valued predicate called *interdatabase dependency predicate* which specifies the relationship between the source and target data objects, and evaluates to true if satisfied. For relational databases, P can be specified, using operators of relational algebra (selection, projection, join, union, difference, intersection, etc.). Together with the basic operators, we also use the aggregate operator ξ and the transitive closure operator α.
- C is a boolean-valued predicate, called *mutual consistency predicate*. It defines consistency requirements involving time and state of data, and specifies when P must be satisfied.
- $\mathcal{A}$ is a collection of *consistency restoration procedures* that specify actions to be taken to restore consistency. The *execution mode* of the action component specifies the way by which a restoration procedure (child) is related to its parent transaction. *Non-vital* specification allows relaxation of atomicity, and *decoupled* specification is used to relax isolation.

The dependencies are specified in a declarative fashion and will be viewed as separate schema entities, collectively referred to as *interdependency schema* (IDS). In other words, the IDS is a set of all D^3s to be enforced in the multidatabase system for managing consistency of the corporate data. They can be used to perform updates in a way that allows us to overcome the limitations imposed by traditional transaction management systems (e.g., two-phase

*Author's work was supported by Bellcore.

†Notice of Disclaimer: The discussion of systems, databases and interdependencies in this presentation is for illustration only. Bellcore makes no representation or warranty, expressed or implied, with respect to the sufficiency, accuracy, or utility of any information or opinion contained herein. Bellcore expressly advises that any use of or reliance upon said information or opinion is at the risk of the user and that Bellcore shall not be liable for any damage or injury incurred by any person arising out of the sufficency, accuracy, or utility of any information or opinion contained herein.

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0483...$1.50



TIRKS is a registered trademark of Bellcore

Figure 1: Examples of Interdependent Data among some Bellcore Systems

commit). For example, we may allow data to be temporarily inconsistent, within specified limits.

In this paper we report the results of a case study to understand applicability, adequacy and advantages of our framework. We studied several systems (existing or currently under development) at Bell Communications Research, Inc. (Bellcore) and we identified examples of actual and realistic interdependencies among them. This empirical study shows that the D^3 framework allows precise and detailed specification of complex interdependencies that may lead to more sophisticated and efficient enforcement of strategies and better support of business rules related to consistency of corporate data. We believe that our framework provides both information system developers and users with many advantages, such as: specification of development requirements of products that meet a business's data consistency needs, enforcement of interoperability of application systems, elimination of time consuming and costly manual operations, and better quality of data to end users. Lack of data consistency is one of the most important problems in today's industries. We hope our work can motivate researchers and vendors to develop solutions to this important problem within the constraints of existing environments.

2 Example Specification of D^3s

The study included many meetings, with product requirements analysts and system developers who provided information relevant to the study, and a consideration of ten systems. Most of the discussions involved systems from the Planning and Engineering area of Bellcore, and their interactions with the source or fundamental data systems (i.e., provisioning and unit inventory systems) and reference data systems. Due to space limitation, we give examples of only two of the interdependencies we studied and documented. Additional examples and detailed discussion can be found in [KS92]. Figure 1 shows a high level view of the systems that featured in the discussions.

Example 1

The relation *DMD_CAP* in PVI (Planning View of Inventory) contains various counts of demand and capacity for an instance of a Planning Unit. The attribute *spare* in table DMD_CAP contains the spare capacity of an inventory item, based on some measurement unit. This information is also kept in the LEIS System, in table CSA_SPARE under the *pots_spr_npl* attribute. The D^3 specifies the consistency requirement (discussed in detail below) between the spare capacity of an inventory item in LEIS and the spare capacity of the same item in PVI. The relevant schema of $PVI.DMD\_CAP$ is:
DMD_CAP
(pln_unit_id, dmd/cap_meas_unit, load_time_period, location_id, total, assigned, spare, tradeable) and $LEIS.CSA\_SPARE$ is defined as:
LEIS.CSA_SPARE (csa, csa_flag, dss_flag, dss_spr_pl, pots_spr_npl, avl_pg_cap, spr_fibers)

The attribute pots_spr_npl in table *CSA_SPARE* is the

source object which must equal the attribute *spare* in table DMD_CAP of PVI, under the conditions *pln_unit_type = 'all dlc sys'* and *dmd/cap_meas_unit = 'npl channel'*. For these related objects, we can allow the target object to diverge with respect to the source object (a) by up to 6 months (defined by c_1), or (b) until a planner explicitly requests an Inquiry of the most recent data for a particular carrier service area – *csa* (defined by c_2). In usual circumstances, a restoration procedure $Update\_PVI.DMD\_CAP$ will be executed once every six months to restore the dependency as required by c_1. It will involve data for all *csa* the LEIS system maintains. Today, this restoration involves a bulk upload. When a planner wants most recent data for a specific *csa* of interest that is supported by c_1 before s/he begins the planning activity, s/he issues the $Planning\_Inquiry$. This is particularly useful when a planner knows or suspects that a particular *csa* has seen much higher than usual recent provisioning activity. Procedure $Update\_PVI.DMD\_CAP\_csaOnly(csa)$ will involve consistency restoration for interdependent data extensions related to the *csa* specified in the $Planner\_Inquiry(csa)$ only. This restoration should be done in a *coupled* and *vital* mode since the planning activity can proceed only after its successful execution.

D_1^3 ::
$\mathcal{S}$:*LEIS.CSA_SPARE.pots_spr_npl*
U :*PVI.DMD_CAP.spare*
P: $\Pi_{csa,pots\_spr\_npl}(LEIS.CSA\_SPARE) =$
$\Pi_{location\_id,spare}$
$\sigma_{pln\_unit\_type='all\ dlc\ sys' \wedge dmd/cap\_meas\_unit='npl\ channel'}(PVI.DMD\_CAP)$
C: $c_1 \vee c_2$ *where*
$c_1 := \epsilon(6\ months)$
$c_2 :=!Planner\_Inquiry(csa)$
$\mathcal{A}$: **when** c_1 *Update_PVI.DMD_CAP* **as** $non-coupled$
otherwise *Update_PVI.DMD_csaOnly(csa)*
as *coupled* **and** *vital*

Example 2

There is a number of cases in the planning system where one system contains an itemized and detailed description of an inventory unit and another system contains only the total number of these inventory units of the same type. One such case occurs between tables in PVI and LSD&F (Local Switching Demand & Facility): $LSD\&F.ENTITY\_JOB$ and $PVI.DMD\_CAP$.
$LSD\&F.ENTITY\_JOB$ is defined as:
ENTITY_JOB (entity, entity_cut, ent_ln_cap, eq_type, job_num, lncap_in_nal, lncap_in_lns)

LSD&F contains detailed information about line capacity in network access lines *(lncap_in_nal)* for every equipment connected to a particular switch, identified by its CLLI-code. On the other hand, PVI contains a total of the line capacity for all switches. The dependency constraint identified above may not hold after an update to the LSD&F database, such as a recent job that connected more cable lines to a switch with a specific CLLI-code, e.g, 12345678. Still PVI can tolerate such updates in LSD&F up to a point specified in the consistency predicate C before it must be made consistent with the related data in LSD&F. Assuming that LSD&F keeps versions of the *ENTITY_JOB* table, PVI can lag behind, up to 3 versions. In addition, if 15% or more of the tuples of the particular switch in *ENTITY_JOB* have been updated, then the consistency should be restored immediately. We can formulate the above interdependency as follows:

D_2^3 ::
$\mathcal{S}$:*LSD&F.ENTITY_JOB.lncap_in_nal*
U :*PVI.DMD_CAP.total*
P: $\Pi_{DMD\_CAP.total}$
$\sigma_{DMD\_CAP.pln\_unit\_id=\&\texttt{CLLI-code}}$
$(DMD\_CAP) =$
$\xi_{Sum(lncap\_in\_nal)}\sigma_{entity=\&\texttt{CLLI-code}}$
$(ENTITY\_JOB)$
C: $c_1 \vee c_2$ *where*
c_1 := *3 versions (ENTITY_JOB) and*
$c_2 := 15\%\Pi_{lncap\_in\_nal}\sigma_{entity=\&\texttt{CLLI-code}}$
$(ENTITY\_JOB)$
$\mathcal{A}$:**when** c_1 *Update_PVI.DMD_CAP* **as** *non-coupled*
otherwise *Update_Switch_Only* **as** *coupled* **and** *vital*

3 Discussion

With the D^3 framework we were able to model all business requirements related to data consistency that we encountered, thus empirically justifying that our framework is comprehensive. Several examples exhibit complex dependencies in their P predicates (like D_2^3 which involves aggregates). Furthermore, our specification is declarative and completely independent of implementation platform, language, and application code.

From a corporate-wide standpoint, an adequate framework for specifying data interdependency supports accurate modeling of requirements, which in turn leads to improved quality of data, which in turn reduces the need for rework and manual intervention during operations. This results in lower operations costs and faster and accurate response for customers. Accurate modeling of clients' requirements also improve the value of Bellcore's product by capturing information clients need to interoperate their systems with our products, and improve system development by allowing better requirements exchange between requirements analysts and developers. A specification framework such as the one used here provides a vehicle for product requirements analysts to communicate with the clients and with the developers (and vice-versa). We are currently studying the issues of the correctness of specifications. We are also developing near-term and long-term solutions for enforcing the consistency requirements in

heterogeneous multi-system environments.

Acknowledgements

This study would not have been possible without significant support from many product requirement analysts and developers in Bellcore. The list is too long. Work on interdependent data is a result of research collaboration with Marek Rusinkiewicz.

References

[KS92] G. Karabatis and A. Sheth. Specifying Interdependent Data: A Case Study at Bellcore. Annual Review of Communications, National Engineering Consortium, Vol. 46, 1992-93.

[RSK91] M. Rusinkiewicz, A. Sheth, and G. Karabatis. Specifying Interdatabase Dependencies in a Multidatabase Environment. IEEE Computer, December 1991.

[SR90] A. Sheth and M. Rusinkiewicz. Management of Interdependent Data: Specifying Dependency and Consistency Requirements. *Proc. of the Workshop on the Management of Replicated Data*, Houston, TX, November 1990.