

Paper

Requirements Blueprint and Multiple Criteria For Distributed Database Design

Ambrose Goicoechea, Ph.D.
Software Engineering Center (SWEC), MITRE Corporation
E-mail: agoico@mitre.org

Paper presented at the International Council on Systems Engineering (INCOSE) Meeting
6-8 April 2000, Reston, Virginia

Requirements Blueprint and Multiple Criteria For Distributed Database Design

Ambrose Goicoechea, Ph.D.
Software Engineering Center (SWEC), MITRE Corporation
agoico@mitre.org

Abstract. The purpose of this paper is two-fold: (1) to present a methodological framework for the design of database architectures in a distributed database environment, and (2) to provide support for the Defense Information Systems Agency's (DISA) guidelines in the segmentation of databases for use by DoD systems and programs. The first problem considered addresses design options in the partitioning of a single large database into multiple, smaller database segments. The second problem considers multiple data sources and determines a preferred subset of these to form a single, composite database while satisfying requirements and multiple criteria. The structured approach considered here is that of non-linear, zero-one mathematical programming (MP) to yield insight into the choice of designs possible given a set of system requirements and criteria.

Keywords. database design, distributed database environment, virtual database design, database segment design, DII COE, Multiple Criteria Decision Making, MCDM.

INTRODUCTION

Increasingly today, many information systems are being designed where the need exists to access multiple bodies of data available not in a single data source (i.e., a database) but in multiple data sources distributed over many programs and systems within the Department of Defense (DoD) community. This is especially apparent when creating database "segments" for the Defense Information Infrastructure Common Operating Environment (COE). The DII COE Integration and Runtime Specification (I&RTS) defines a segment as a collection of one or more software or data units most conveniently managed as a unit. A database segment is data that is to be managed by a Data Base Management System (DBMS) host on a DII COE server. This paper addresses the following questions:

- What are the options available to the database designer as he/she considers multiple data sources? Should a "preferred" or optimal set of data sources be identified?

- If database segments are to be designed by contractors for the DII COE, what techniques and guidelines can be generated for database "segmentation" ?
- How are segments to be integrated to create an information system? Should the segmentation process "encourage" the production of many small segments or just a few large segments to create such an information system?

The motivation for these questions stems from the DII COE I&RTS (Version 4.0, October 1999).

DoD's Vision. "The DII COE originated with a simple observation about command and control systems: certain functions (mapping, track management, communications interfaces, etc.) are so fundamental that they are required for virtually every command and control system. Yet these functions are built over and over again in compatible ways even when the requirements are the same, or vary only slightly, between systems. If these common functions could be extracted, implemented as a set of extensible low-level building blocks, and made readily available to system designers, development schedules could be accelerated and substantial savings could be achieved through software reuse. Moreover, interoperability would be significantly improved because common software is used across systems for common functions, and the functional capability only needs to be built correctly once rather than over and over again for each project".

The DII COE Concept. Both software and data reuse are encouraged to achieve interoperability. Principles that are part of this concept include (quote):

- An architecture and approach for building interoperable systems,
- An environment for sharing data between applications and systems,
- An infrastructure for supporting mission-area applications,
- A collection of reusable software components and data,
- A rigorous set of requirements for achieving DII

compliance,

- An automated tool set for enforcing COE principles and measuring DII compliance,
- An approach and methodology for software and data reuse, and
- A set of Application Program Interfaces (APIs) for accessing COE components.

Database Segment Development. The DII COE document does contain general guidelines that database segment developers can use to meet data requirements (quote):

- Use existing data stores at runtime (this requires acquisition of access rights). Data stores can be used as-is or changes can be negotiated with the data store owner.
- Review the database segments on the COE Data Emporium for potential reuse. For example, if the Emporium contains a database segment that contains the common representation for Organizations, then that segment should be used.
- Attempt to have existing segments updated to satisfy the new requirements. Use similar enterprise segments (preferred) or Community of Interest segments if available.
- Create new segments to extend the existing segment(s) as necessary to meet the requirements.
- Create a new data segment by reusing as much existing schema as possible and augmenting it to meet the requirements.

Other factors are recommended as well towards determining the structural contents of a database (quote):

- Which tables can be conveniently managed as a unit.
- Which tables are defined to support a functional area.
- What are the sources of data, and
- What are the database object dependencies.

"The advantage of multiple shared database segments is that the segments are more granular, therefore allowing a shared data server to be configured to support mission applications without having to carry superfluous data and can be handled as separate configuration items. A disadvantage of multiple shared database segments is the management of database object dependencies that can be created by such things as foreign key constraints. These inter-segment dependencies complicate the management of

segment installation and, moreover, the removal of segments." These guidelines, however, do not offer specific ways or mechanisms that a designer can follow to select a preferred subset of data sets or to partition a database into segments while minimizing the number of foreign-key dependencies.

STATEMENT OF PROBLEM

This paper addresses two specific distributed-database design problems:

Problem 1: Given a determination to partition a database into database segments (i.e., subsets of tables), what are the design options available to the developer, and what are the trade-offs involved in terms of cost, data sharing, interoperability, performance, and other criteria? The designer begins with one large database and proceeds to partition it into several smaller databases or segments, i.e., a "one-to-many" database segmentation problem.

Problem 2: Given multiple, alternative data sources, how does the database designer select an optimal subset of data sources? Part of the system design may be the formation of an "integrated, virtual database" or a system interface that enables queries of the various data sources. If query performance (i.e., non-functional requirement) is important then the design solution may identify a particular set of data sources. If keeping system development costs down is a primary concern then the design solution may identify a different set of data sources and this design will not necessarily yield high query performance. The designer begins with a large collection of databases and proceeds to select portions of each database to form a single, composite database, i.e., a "many-to-one" database segmentation problem.

METHODOLOGICAL APPROACH

Mission requirements drive the selection of data sources, database design options reflect alternative solutions in a "solutions-space" (i.e., one single, large database system; several small databases; single site; multiple site, distributed database environment; etc.), while development costs and non-functional requirements (i.e., system performance, security, reliability, etc.) contribute to identifying "a preferred system", as depicted in Figure 1. See Ozsu and Valdiriez (1991) and Simon (1995) for a presentation of design issues and a preview of emerging technologies in distributed database design.

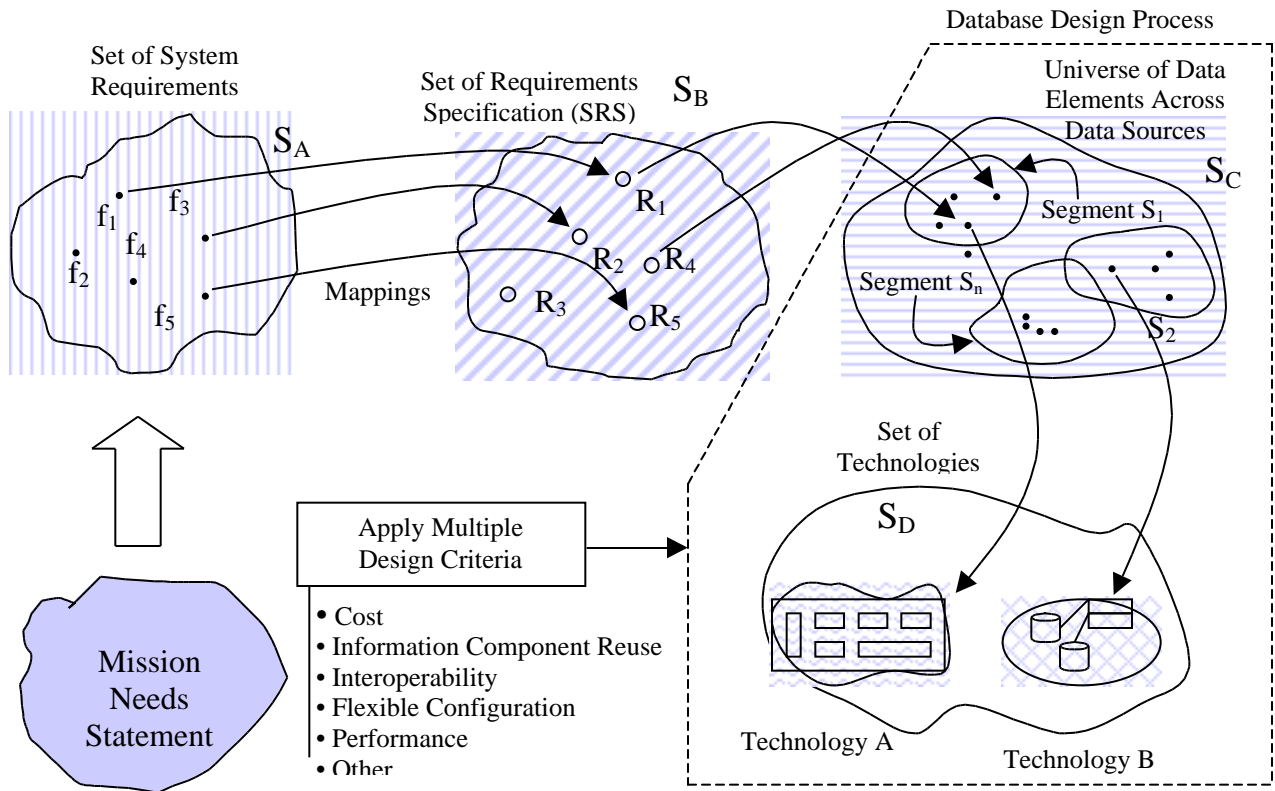


Figure 1. Mapping of System Requirements to Data Elements in a Distributed Database Environment this possible maximum number N_2 is:

Problem 1: One-to-Many Database Segmentation.

An approach suggested here is to partition a large database (i.e., a set S of tables) into multiple segments (i.e., a database segment S_i is made up of a subset of the tables in S , such that $S_i \subset S$) so that each table appears in one and only one segment and the union of all segments is the set S , as depicted in Figure 2 for the general case of n data sources:

- (a) Consider now the specific example of a database to be segmented as shown on Figure 3. A 4-table segment candidate in Figure 3 has 6 foreign-key dependencies, i.e., 6 segment boundary “crossings”; an arrow points to the child-end of a relationship between two tables; the other 4-table segment also has 6 dependencies, and the 3-table segment has 8 dependencies; however, counting the total number of crossings across all three segments yields a total of $N_1 = 10$ foreign-key dependencies.

- (b) Formulation of the “segmentation design problem” shows that the number of possible pairwise dependencies (i.e., at least one foreign key is involved between two tables) is the number of combinations given by the *binomial coefficient*. For the example database in Figure 3

$$\binom{10}{0} = \frac{10!}{0!9!} = \frac{(10)(9)!}{9!} = 45$$

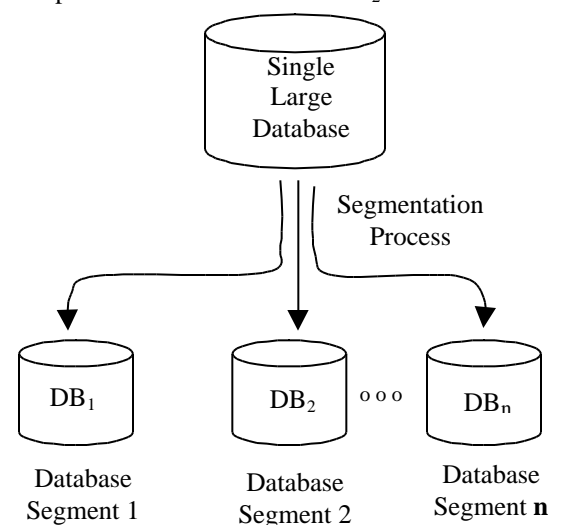


Figure 2. Partitioning of a Database into n Segments

However, the actual number N_3 of table pairs with dependencies is only 15:

A-B	D-K	C-I
A-I	E-F	C-D
A-F	E-I	C-G
B-K	F-H	I-C
B-D	H-G	I-E

- (c) The number of “segment dependencies” N_1 was shown to be 10 which is smaller than the

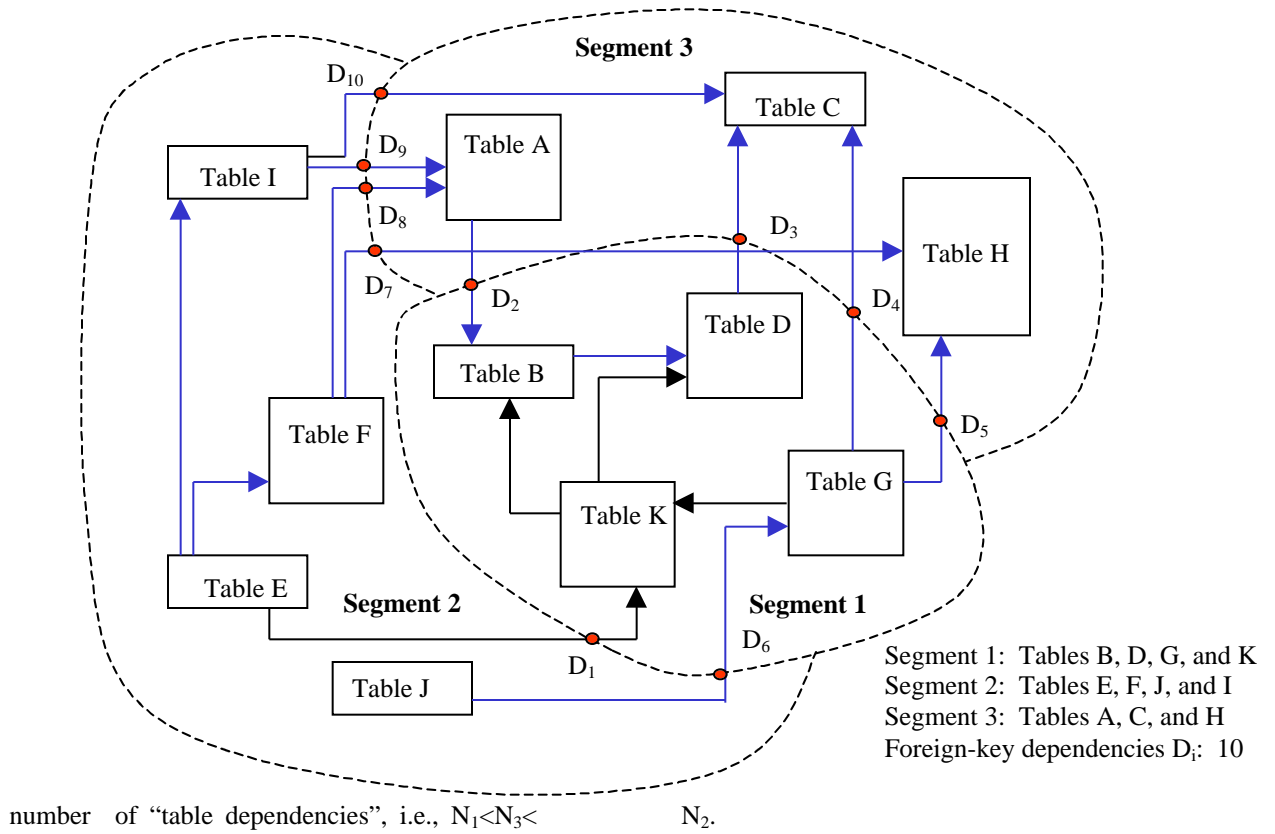


Figure 3. Candidate Database Segments with 10 Foreign-Key Dependencies

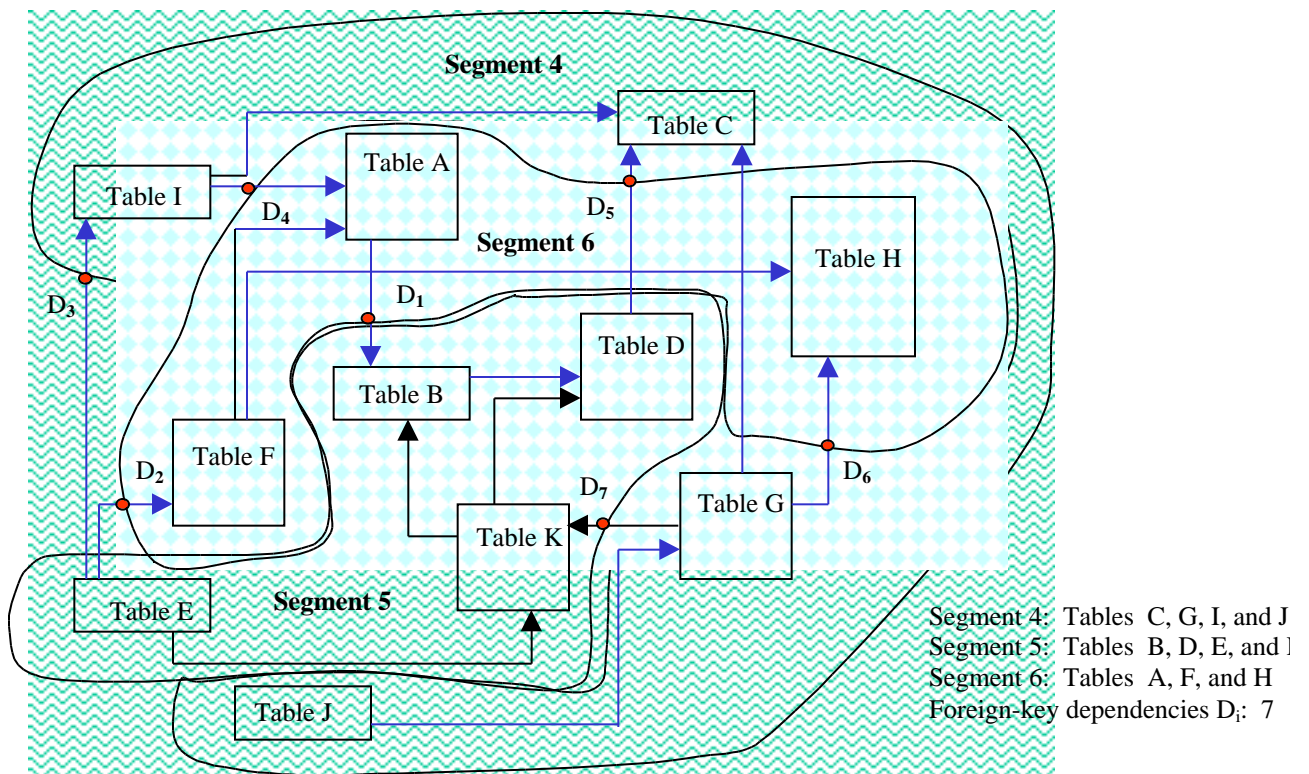


Figure 4. Optimal Segmentation of Database Resulting in Smallest Number (7) of Foreign-Key Dependencies.

(d) Number of segments possible, 3 or 4 tables each:

$$\frac{10}{4} = \frac{10!}{6!4!} = \frac{(10)(9)(8)(7)}{(4)(3)(2)} = 210$$

$$\frac{6}{3} = \frac{6!}{3!3!} = \frac{(6)(5)(4)}{(3)(2)} = 20$$

Out of the set of 10 tables, for example, 210 segments could be constructed with 4 tables each; of the remaining 6 tables 2 segments could be constructed with 3 tables each; the total number of possible designs (i.e., one design consisting of one 4-table segment and two 3-table segments) would be $(210)(20) = 4,200$, a very large number of segment designs indeed.

Mathematical Formulation:

Let $X_{ji} = 1$ if table i belongs to segment j , 0 if table i does not belong to segment j , and such that $i = A, B, \dots, K, L$, and $j = 1, 2, \text{ and } 3$ as depicted in Figure 3. Segments 1, 2, and 3, for example, have a total of 10 foreign-key dependencies. Then, the optimization problem can be stated as follows:

Minimize:

$$X_{ij}X_{kl}$$

i, j, k, l

where $k, j = A, B, C, \dots, L$, the names of tables, but $k \neq j$; also $i, l = 1, 2, \text{ and } 3$, the names of the segments, but $i \neq l$, subject to constraints:

$X_{1A} + X_{1B} + X_{1C} + \dots + X_{1L} = 4$ to require only four tables in segment 1; $X_{2A} + X_{2B} + X_{2C} + \dots + X_{2L} = 4$ to require only four tables in segment 2; $X_{3A} + X_{3B} + X_{3C} + \dots + X_{3L} = 3$ to require only 3 tables in segment 3; $X_{1A} + X_{2A} + X_{3A} = 1$ to require that table A belong to one segment only (either segment 1, 2, or 3); $X_{1B} + X_{2B} + X_{3B} = 1$ to require that table B belong to one segment only; $X_{1C} + X_{2C} + X_{3C} = 1$ to require that table C belong to one segment only; and so forth for all other remaining tables; also, and $X_{ij} = 1$ or 0 for all i and j .

Solution:

The optimal solution to this non-linear, binary problem was obtained using the mathematical programming capability in Excel's Solver tool:

$X_{1C} = 1$	$X_{1G} = 1$	$X_{1I} = 1$
$X_{1J} = 1$	$X_{2B} = 1$	$X_{2B} = 1$
$X_{2D} = 1$	$X_{2E} = 1$	$X_{1K} = 1$
$X_{3A} = 1$	$X_{3F} = 1$	$X_{3H} = 1$

and all other variables are zero, i.e., do not select. A graphical representation of this solution is shown in Figure 4. This solution, it is seen, produces segments 4, 5, and 6 with the smallest possible number of 7 foreign-key dependencies. This mathematical representation can now be applied to the general problem with hundreds or thousands of tables, any segment size desired, and can also accommodate other constraints such as week entities, business rules, and groups of tables that the user wishes to be contained within the same segment.

Problem 2: Many-to-One Database Segmentation.

As the database designer sets to identify existing multiple databases that may contain the tables and data elements of interest to him/her several situations are possible, as depicted on Table 1. In the simplest case, Case 1, all desired data elements or objects (i.e., data elements, tables, stored procedures, other) reside in one data source so the design strategy is to utilize that single data source and no decision opportunity exists. In Case 2 all desired objects reside in multiple copies of the same data source. In Case 3 desired objects reside across multiple, distinct data sources, with no sharing of data objects across data sources. Finally, in Case 4 desired objects reside across multiple, distinct data sources, but some data objects are shared across data sources available. We proceed to consider a decision problem in Case 4.

Selection of the Preferred Subset of Data Sources.

Next, we consider an illustrative example of Case 4 where there are a total of 7 desired data elements, $\{1, 2, 5, 7, 9, 10\}$, distributed over 5 data sources as shown on Figure 5 that are needed to meet requirements, with cost and query response times as shown on Table 2.

A decision point occurs given the opportunity to select a "preferred subset" of n data sources that meets the following criteria:

- (1) all the desired data elements are represented in this preferred subset,
- (2) it contains the smallest number of data sources needed to provide all the data elements in the "design subset of data elements", as in Figure 1,
- (3) it belongs to the set of non-inferior solutions on the *Pareto design frontier*. This frontier can be obtained by considering multiple criteria (e.g., minimize design cost, minimize aggregate query response, other) within a multiple-criteria decision making (MCDM) framework (Goicoechea et al., 1992, 1982).

Visual inspection of this illustrative example reveals three possible solutions, i.e., designs:

- Design 1: Choose data sets: A + B + C
- Design 2: Choose data sets: A + D
- Design 3: Choose data set: E

That is, solution/design 1 yields a choice subset consisting of data sources A, B, and C; solution 2 yields a choice subset consisting of data sources A and D; and solution 3 yields a choice subset consisting of data sources E only. We note that each of these three solutions contain the entire design set of data elements, {1,2,5,7,9,10}.

In a real-world problem the number or alternative data sources may be large and simple enumeration of the possible solutions may not be practical. Mathematical

programming (MP) is one approach to solving this design problem, as shown next.



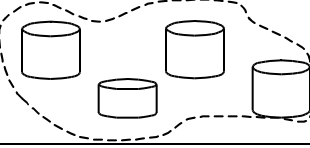
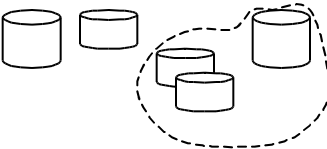
Multiple Criteria

Multiple criteria in distributed database design can include:

- Cost
- Performance (query response time, other)
- Reuse/sharing of data sources
- Flexibility of configuration

Goicoechea et al. (1992, 1982) describe the use of multiple criteria in engineering design and business decision problems.

Table 1. A Distribution Pattern of Data Elements Across Multiple Data Sources

Case No.	Distribution of Available Data Elements	Architectural Strategy	Optimization Criteria
1	All desired data elements/objects reside in one data source.	 Select single data Source	None. No decision opportunity exists.
2	Desired data objects reside in one data source with exact replications in multiple sites.	 Select single data source	Select site that offers lowest system development cost. Other criteria: - performance
3	Desired data objects reside across multiple, distinct data sources; no sharing of data objects across data sources.		Select entire set of data sources. No decision opportunity exists.
4	Desired data objects reside across multiple, distinct data sources; there is sharing of data objects across data sources.	 Select subset of data sources	Select subset of data sources using optimization criterion: - Performance - Cost - Other

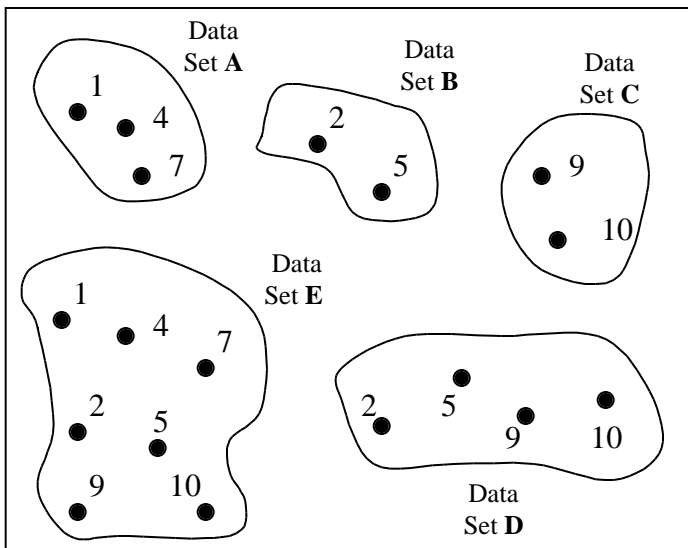


Figure 5. Alternative Multiple Data Source

A design variable is assigned to each data element in the “choice sub-set of data elements” with a range of possible values of 1 (i.e., select this data element and its data source) and 0 (i.e., do not select this data element and its data source).

Design variables: Let

- X_{ij} = data element j in data source i ,
- X_{A1} = data element 1 in data source A
- X_{A4} = data element 4 in data source A
- X_{A7} = data element 7 in data source A
- X_{B2} = data element 2 in data source B
- X_{B5} = data element 5 in data source B
- X_{C9} = data element 9 in data source C
- X_{C10} = data element 10 in data source C

X_{D2} = data element 2 in data source D
 X_{D5} = data element 5 in data source D
 X_{D9} = data element 9 in data source D
 X_{E2} = data element 2 in data source E
 X_{E4} = data element 4 in data source E
 X_{E5} = data element 5 in data source E
 X_{E7} = data element 7 in data source E
 X_{E9} = data element 9 in data source E
 X_{E10} = data element 10 in data source E

X_{D10} = data element 10 in data source D
 X_{E1} = data element 1 in data source E

$X_{A1}, X_{A4}, X_{A7}, \dots, X_{E10} = 0,1$ all decision variables are binary (zero or 1 values).

The solution to this illustrative problem was then obtained via mathematical programming (MP) with Microsoft Excel's Solver:

$X_{E1} = 1$ $X_{E5} = 1$
 $X_{E2} = 1$ $X_{E7} = 1$
 $X_{E4} = 1$ $X_{E9} = 1$
 $X_{E10} = 1$

and all other variables were determined to be zero, thus identifying data source E as the choice data set, which corresponds to Design 3 shown earlier. However, if the design criterion is to minimize aggregate query response time across all data sources to be selected (i.e., maximize system time performance) then new MP solution is given by data sources A +B +C, which corresponds to Design 1 and which can also be verified by inspection of Table 2. Next, we proceed to apply both cost and performance criteria:

Minimize: $F = w_1 F_{\text{cost}} + w_2 F_{\text{performance}}$
 $= w_1 [C_1 X_{A1} + C_2 X_{A4} + \dots + C_{18} X_{E10}]$
 $+ w_2 [R_1 X_{A1} + R_2 X_{A4} + \dots + R_{18} X_{E10}]$

subject to the same constraints since the "design space" remains the same. The solution to this multi-criteria design problem with *criterion weights* $w_1 = 0.6$ and $w_2 = 0.4$ yields:

$X_{A1} = 1$ $X_{D2} = 1$ $X_{D10} = 1$
 $X_{A4} = 1$ $X_{D5} = 1$
 $X_{A7} = 1$ $X_{D9} = 1$

which corresponds to Design 2. Other sets of weights can be tried such that $w_1 + w_2 = 1.0$ until all non-inferior solutions in the Pareto frontier have been identified. In our example there are only three possible designs, and these are now presented on Table 3 and Figure 6.

Table 3. Multiple Design Criteria and Database Designs

Criteria	Design 1	Design 2	Design 3
1. Cost (\$)	31	19	12
2. Performance (Total Query Time, Sec.)	21	23	120

Table 2. Cost and Query Time

Design Variable	C_i Cost (\$,Dollars)	R_i Query Response Time (Sec)
X_{A1}	2	4
X_{A4}	2	4
X_{A7}	3	3
X_{B2}	10	3
X_{B5}	12	3
X_{C9}	1	2
X_{C10}	1	2
X_{D2}	3	3
X_{D5}	3	3
X_{D9}	3	3
X_{D10}	3	3
X_{E1}	2	15
X_{E2}	2	20
X_{E4}	1	15
X_{E5}	2	20
X_{E7}	2	20
X_{E9}	2	15
X_{E10}	1	15

Minimize overall system cost function:

$F_{\text{cost}} = C_1 X_{A1} + C_2 X_{A4} + C_3 X_{A7} + C_4 X_{B2} + \dots + C_{19} X_{E10}$

Subject to the following constraints. $X_{A1} + X_{A4} + X_{A7} + X_{B2} + \dots + X_{E10} = 7$

the total number data elements is 7 as needed to meet system requirements;

$(X_{A1} + X_{A4} + X_{A7})/3 = 0,1$

binary constraint, i.e., select all or none of the data elements in data source A;

$(X_{B2} + X_{B5})/2 = 0,1$

binary constraint, i.e., select all or none of the data elements in data source B;

$(X_{C9} + X_{C10})/2 = 0,1$

binary constraint, i.e., select all or none of the data elements in data source C;

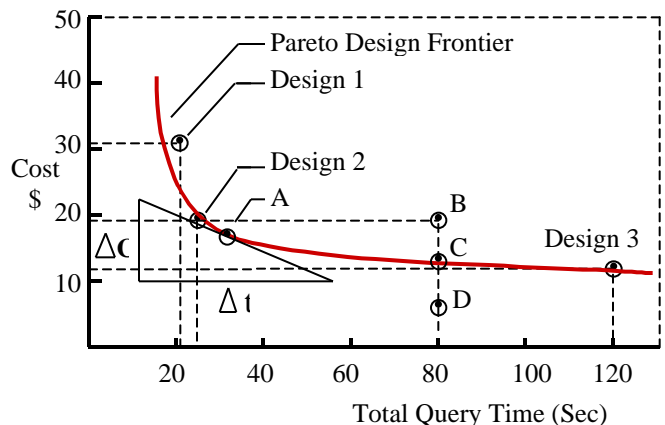


Figure 6. Pareto Frontier in Database Design

Several observations are made. First, the optimization procedure above generated the three possible solutions/designs in this illustrative example, and these solutions make up the *Pareto design frontier*. The DB designer aims at obtaining designs that are *Pareto-optimal*, that is designs that are part of the Pareto design frontier. Solutions over this frontier are such that as one moves from Design 1 to Design 2, the cost decreases from \$31 to \$19 at the expense of query performance (i.e., total query time increases from 21 sec. to 23 sec.). A tangent to the design frontier at point A is C/t , which is the *criteria tradeoff* at that point; at point A, for example, an improvement in query performance in the order of 20 seconds costs \$10, approximately.

It would not be immediately apparent to the DB designer working with say hundreds of data sources, whether he/she is obtaining DB designs that are "above", "below", or "on top" of the Pareto frontier, and it then becomes necessary to obtain and compare several solutions. Solution B, for example, is "inferior" to Solution C because whereas both offer the same performance, solution C has a lower cost. Solution D has still a lower cost but is not technologically possible. As the designer maintains performance at a fixed level and continues to make design changes each lowering the total, eventually he will not be able to make further designs changes that translate into a lower cost. It is at that point that he has reached the Pareto design frontier. Each technology (e.g., flat file database design and operation, Oracle DBMS, Sybase DBMS, etc.) has its own Pareto design frontier.

CONCLUSION

Database segmentation presents many challenges to the database engineer in his/her efforts to satisfy system requirements while applying multiple design criteria. Minimizing server administrative overhead, cost, number of table joins needed to execute a query, and physical distance to data source, for example, is desirable. On the other hand, maximizing data source availability, database performance (through record indexing at the data source, efficient schema design, efficient reference data set design, containment of database fragmentation, other) is desirable. Identification of these often conflicting criteria, relationships among them, and providing the means to measure criteria values of alternative designs early in the database design phase should yield a positive payoff later during database development and eventual field deployment and operation.

Systems engineering can provide a framework for effective analysis of requirements, selective utilization of optimization techniques, and multiple criteria tradeoff analysis of alternative database designs.

The first part of this paper showed that it is possible to consider the general design problem of partitioning a large database into a desired number of database segments in such a way that the number of foreign-key dependencies is the smallest number possible. This smallest number of foreign-key dependencies can have a beneficial impact on overall system performance, including a decrease in database server administrative overhead. This approach can now be applied to large databases containing hundreds of tables.

The second part of this paper showed how to apply multiple-criteria decision making (MCDM) techniques to virtual database design. Hundreds and possibly thousands of data sources may be initially available to the database designer as he/she searches for the best selection of a relatively small number of data sets. This approach can guide the database designer in the search of Pareto database designs (best selection of data sources) that feature desired, multiple criteria levels and tradeoffs.

Topics for future research applying the problem representation and solution techniques demonstrated above include data sharing and system inter-operability in a multiple-project, multiple-client distributed database environment.

Acknowledgements

The author wishes to thank Dr. Robert M. Daniels, Jr., and Mr. Francis B. Driscoll, Jr., MITRE, Software Engineering Center, W098 for their review of this paper and valuable comments.

Bibliography

1. *Defense Information Infrastructure (DII) Common Operating Environment (COE), Integration and Runtime Specification (I&RTS)*, Defense Information Systems Agency (DISA), Report, Revision 4.0, October 1999.
2. *Principles of Distributed Database Systems*, by M.T. Ozsu and P. Valdiriez, Prentice Hall, 1991.
3. *Multiple Criteria Decision Making (MCDM), Applications in Industry, Business, and Government*, by A. Goicoechea, L. Duckstein, and S. Zions (editors), Proceedings of the IX-th International Conference on MCDM, Fairfax, Virginia, August 5-8, 1992.
4. *Multi-objective Decision Analysis with Engineering and Business Applications*, by A. Goicoechea, D.R. Hanson, and L. Duckstein, John Wiley and Sons Publishers, 560 pages, New York, 1982.
5. *Strategic Database Technology: Management for the Year 2000*, by A.R. Simon, Morgan Kaufmann Publishers, 1995.

Biographical Sketch

Dr. Ambrose Goicoechea has over 15 years of experience in the design and application of systems engineering principles and methodological frameworks to problems in architecture design, decision support system design, database design, transportation systems, data fusion for object classification, and design of multiple criteria decision support systems (MCDSS). In April 1999 he joined the Software Engineering Center (W098) as a Sr. Information Systems Engineer and he has recently completed an evaluation of the messaging middleware (MQSeries) infrastructure at U.S. CUSTOMS and recommended its adoption in the IRS modernization architecture. At W098 he also completed an architecture design to support intra-site and inter-site data backup, failover, and recovery for the Army's Defense Message System (DMS); server cluster software and technologies from five major vendors were evaluated with recommendations for an augmented DMS architecture. Prior to joining MITRE he was the Performance and Capacity lead engineer for the Global Transportation Network (GTN) at Lockheed Martin Corporation, in Manassas, Virginia (1997-1999) where he created the system performance and capacity planning (PCP) group, designed and instituted PCP processes, responsible for setting up a suite of modeling and measurement tools. Associate Professor in the Systems Engineering Department, George Mason University, 1985-1999. As President and Technical Director of Integrated Technologies and Research, Inc., from 1995 to 1997, he developed decision support systems for the U.S. Army Corps of Engineers. In 1985 and 1986 he was NASA-ASEE Research Fellow at the Goddard Space Flight Center, in Greenbelt, Maryland, designing decision support systems for NASA managers to assist with systems engineering and configuration functions of space projects; also, a member of the Man-Machine Interface Design Group. 1979, NASA-ASEE Research Fellow, Jet Propulsion Laboratory of the California Institute of Technology (Cal-Tech); evaluation and selection of projects in the areas of solar-thermo power plants, underground nuclear plant location analysis, and urban public transportation systems. Organizer and General Chairman of the IX-th International Conference on Multiple Criteria Decision Making (MCDM), Fairfax, Virginia, August 5-8, 1990. Dr. Goicoechea is a recognized speaker at international conferences on system performance modeling, decision analysis, distributed database design, and risk analysis. A member of several engineering professional organizations, and a lecturer and instructor in the Department of Engineering Management, Department of Operations Research, and the Department of Management Science of George Washington University, 1990-Present. He is the author of over 20 published papers and 3 technical books used in engineering, mathematics, economics, and business schools in over 30 universities in the USA and other countries in Latin America, Asia, and Europe.

