computer-aided database design

the DATAID project

edited by a. albano, v. de antonellis and a. di leva

north-holland

COMPUTER-AIDED DATABASE DESIGN
The DATAID Project

Edited by

Antonio ALBANO
Università di Pisa
Italy

Valeria DE ANTONELLIS
Università di Milano
Italy

and

Antonio DI LEVA
Università di Torino
Italy

1985

NORTH-HOLLAND
AMSTERDAM • NEW YORK • OXFORD

ISBN: 0 444 87735 5

Published by:
ELSEVIER SCIENCE PUBLISHERS B.V.
P.O. Box 1991
1000 BZ Amsterdam
The Netherlands

Sole distributors for the U.S.A. and Canada:
ELSEVIER SCIENCE PUBLISHING COMPANY, INC.
52 Vanderbilt Avenue
New York, N.Y. 10017
U.S.A.

PRINTED IN THE NETHERLANDS

PREFACE

At present, database is an important area of study in computer science. On the one hand, it is in itself an area of interest, having produced a large number of results on specific topics, like relational theory, data models, database system architectures, database programming languages, distributed databases, and physical file design. Secondly, it is an area where results of other areas of computer science, such as Programming Languages, Artificial Intelligence and Software Engineering, have been used to produce new interesting applications.

In the last few years, there has been a great interest in database design, where the goal is to combine results from knowledge representation schemas, software engineering, data semantics, relational theory and physical file design in order to define: a) a methodology to go from user requirements to the implementation of applications using DBMSs, through a sequence of structured steps; b) an environment of automated tools to assist the designer during the development process: documentation tools, graphical interfaces, mapping tools between specifications given at different levels of abstraction.

This book addresses both the aspects of database design, and presents a methodology, for centralized and distributed databases, together with a set of automated tools to support the design process. The book itself evolved from the experience of researchers from different Italian universities, industries and research institutions participating in a joint 5-year project, called DATAID, supported by the Italian National Research Council, within the nationwide Progetto Finalizzato Informatica research project.

The book follows a previous one, published two years ago: S. Ceri (ed.), Methodology and Tools for Database Design (North-Holland, Amsterdam, 1983). The latter was mainly concerned with the proposed methodology for centralized databases, although preliminary results regarding the tools under implementation were also included. This book, instead, is mainly dedicated to the description of tools in an advanced stage of implementation and contains the extension of the methodology to distributed databases. We have decided to publish these results because we believe that the DATAID project has achieved significant contributions to the database design field, and the two books contain specific solutions to issues of general interest.

The preparation of the book has covered a period between June and December 1984.

ACKNOWLEDGEMENTS

This book would not have been possible without the support and the encouragement from the direction of the Progetto Finalizzato Informatica, in the persons of Prof. Angelo Raffaele Meo and Dott. Paolo Bronzoni.

THE DATAID PROJECT

DATAID is a 5-year project financed by the Italian National Research Council within the Progetto Finalizzato Informatica. The project started in September 1979 with the participation of several working groups from both the academic and the industrial environments. The aim of the project is the development of a computer aided methodology for database design.

In 1980 a survey was carried out concerning existing methodologies and related tools, and the architecture of a manual methodology, called DATAID-1, was defined.

In 1981 a first version of the methodology was released, and the introduction of automated tools in the design process was considered.

In 1982 the functional specifications of a first set of tools for conceptual design have been produced. For the dissemination of the methodology, several workshops and a first course were held, for which a training manual was prepared. Experimentation of the methodology has been initiated in Local and Central Government applications.

In 1983 and 1984 the methodology has been revised and extended to include distributed database design. At the same time, prototypes of the tools have been developed.

At present, the DATAID working groups are

Universities:

Dipartimento di Elettronica, Politecnico di Milano

Dipartimento di Informatica, Università di Pisa

Dipartimento di Informatica, Università di Torino

Dipartimento di Informatica e Sistemistica, Università di Roma

Istituto di Cibernetica, Università di Milano

Research Centers of Italian National Research Council:

CIOC, Centro di Studio per l'Interazione Operatore-Calcolatore, Bologna

CNUCE, Centro per il Calcolo Elettronico, Pisa

IASI, Istituto di Analisi dei Sistemi ed Informatica, Roma

Industries:

CRAI, Consorzio Calabro per la Ricerca e le Applicazioni nel Settore dell'Informatica, Rende (Cosenza)

CSELT, Centro Studi e Laboratori Telecomunicazioni, Torino

DATABASE Informatica, SpA, Roma

ELEA, SpA, Olivetti Formazione e Consulenza, Firenze

IPACRI, SpA, Istituto per l'Automazione delle Casse di Risparmio Italiane, Roma

Systems & Management, SpA, Torino

Carlo Batini and Antonio Di Leva were coordinators of the project from 1979 to 1981, while the present coordinators of the project are Valeria De Antonellis and Antonio Di Leva.

A list of contributions in journals and of papers presented at conferences as well as technical reports written by project members, from 1979 to 1985, is given in Appendix. Requests for publications may be addressed to the authors.

CONTENTS

COMPUTER-AIDED DATABASE DESIGN: The DATAID Project
A. Albano, V. De Antonellis, and A. Di Leva (Editors)
© Elsevier Science Publishers B.V. (North-Holland), 1985

COMPUTER-AIDED DATABASE DESIGN: THE DATAID APPROACH

Antonio Albano
Dipartimento di Informatica, Università di Pisa
Corso Italia, 40 - 56100 Pisa

Valeria De Antonellis
Istituto di Cibernetica, Università di Milano
Via Viotti, 5 - 20133 Milano

Antonio Di Leva
Dipartimento di Informatica, Università di Torino
Via V. Caluso, 37 - 10125 Torino

The approach to the design of applications using centralized and distributed databases adopted in the DATAID Project is described, together with the role of the automated tools that are to be integrated into a database development environment to support the design methodology.

1. INTRODUCTION

The development of computer technology, and the reduction of its costs, continuously increase the use of DBMSs to maintain the data needed by the functions of an organization. In the early '70s, DBMSs were used to support basic, independent and structured organizational functions. The trend is now toward a global integration of functions, with DBMSs used to support managers in decision making; and in the future DBMSs will be integrated with expert systems to build computerized information systems that will be more effective for decision makers, and able to assist non-expert users in gaining access to data and in interpreting them.

The growing use of DBMSs, the complexity of the new applications, and the need to implement database applications that can be readily adapted to changes in user requirements, has led to an increasing demand for environments with an integrated set of automated tools to support both the design and the maintenance of database applications. The problem is similar to that of software engineering and the following strategies have been suggested: a) the definition of a design methodology composed of a set of structured steps in which design decisions are considered one at a time to obtain a satisfactory design; b) the definition of techniques to be used during the design steps; c) the definition of tools for an automated development support system. Database design techniques are required basically for the following activities:

- Data Analysis, to help the users in organizing their information needs in a structured and stable way to support the evolution of the applications easily.
- Prototyping, to build an early version of the system to be implemented that exhibits the essential features of the future operational system, except for the resource requirements and performance. A prototype helps the users both to determine

whether or not the system in development matches their requirements before the final implementation begins, and to improve their perception of organization needs.

. Implementation, to convert the results of analysis and the prototype into an operative system with a satisfactory storage use and an efficient execution of applications.

The approach to the design of applications using a centralized or a distributed database adopted in the DATAID Project is described, together with an overview of the role of the automated tools under development. Section 2 describes the aspects considered during the database design process, the proposed abstraction mechanisms and the diagrammatic representations used. Section 3 presents the DATAID methodology, and Section 4 describes the automated tools.

2. WHAT TO MODEL AND HOW TO MODEL

Among researchers and practitioners there is general consensus about the basic aspects that should be modeled during the database design process:

. Concrete knowledge, that is, the entities of the observed system, their properties and relationships between them.

. Abstract knowledge, that is, the integrity constraints, which impose restrictions on the legal states of the model of the observed system.

. Procedural knowledge, that is, the basic operations that must be applied to the concrete knowledge so that the model evolves to reflect the changes in the observed system.

. Dynamics, that is, how the concrete and procedural knowledge can be used to model the permissible sequences of events in the organization's functions (or activities), and communication both among activities and activities and users.

The approaches to database design can differ about the methodology adopted in data analysis, the abstraction mechanisms used to model the above aspects, and the features of the language designed to support the abstraction mechanisms during requirements specification and conceptual design /6,7,8/. In the DATAID Project the following alternatives have been investigated.

Concrete Knowledge

Two data models have been used: an Enriched Entity Relationship Model (EERM) and a Semantic Data Model (SDM).

EERM provides the analyst with four abstraction mechanisms: entities, relationships, attributes, and abstraction hierarchies /9/. The differences between the EERM and the ER model introduced by P.P. Chen /11/ are:

Abstraction hierarchies. The first type of hierarchy is the 'subset relationship': an entity E_1 is a subset of the entity E_2 if every occurrence of E_1 is an occurrence of E_2 as well. The second type of hierarchy is the 'generalization': an entity E is a generalization of the entities E_1, ..., E_n, if each occurrence of E is also an occurrence of at most one of the entities E_1, ..., E_n (ISA exclusive hierarchy). The partition over the occurrences of E is established by the value of a property of E (underlying attribute). Both in subset relationships and in generalizations, attributes and relationships of the entity at the upper level of a hierarchy are inherited by the entities at the lower level, which may have additional attributes and relationships. Abstraction hierarchies are 'complete' if for each occurrence of the upper level entity there exists a corresponding occurrence of the entity immediately lower in level.

Aggregates. A set of attributes that can be referred to as a single property.

Repeating attributes. Attributes that can have more than one value.

Identifiers. An internal identifier is an attribute or a group of attributes that determine uniquely an entity; entities may also be identified through other entities associated with them (external identifiers).

The diagrammatic representation of the EERM abstraction mechanisms is shown in Table 1.

SDM provides the analyst with at least the following mechanisms:

Classification. The entities being modeled that share common properties are gathered into classes. The names of the classes denote the elements present in the database. The elements of a class are represented uniquely, that is, only one copy of each element is allowed.

Aggregation. The elements of classes are aggregates, i.e., they are abstractions having heterogeneous components and may have elements of other classes as components. Consequently, relationships among entities are represented by aggregations, rather than with a separate mechanism as in EERM.

Generalization. Classes themselves can be organized in a hierarchy through a partial order relationship, often referred to as the ISA hierarchy: classes in this relationship model entities that play different roles in the observed system, and may be described with a different level of detail. If A is a subclass of B, the following properties hold: a) the elements of A are in every state a subset of the elements of B; b) the elements of A inherit all the properties of the B elements.

Other features of a SDM are: a) it supports multi-valued attributes; b) it allows the definition of procedurally defined information, such as derived attributes or derived classes; c) it provides special syntax for common constraints, such as ranges for attributes, cardinality of relationships, identifying attributes, optional vs. required attributes, modifiable vs. constant attributes /6,14/.

The diagrammatic representation of the abstraction mechanisms is the same as that of EERM, with only this difference, that instead of naming a relationship in a diamond, in the case of SDM the names of the attributes used to model the relationship with the aggregation are shown.

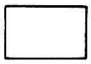
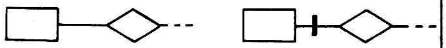
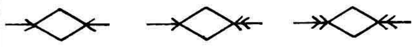
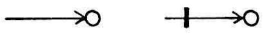
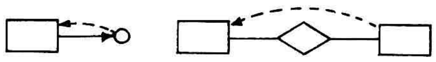
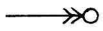
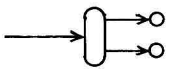
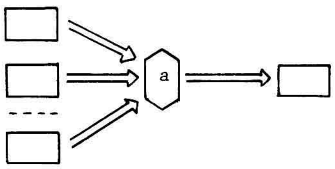
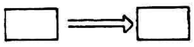
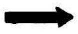
CONCEPT	REPRESENTATION
ENTITY	
RELATIONSHIP (total/partial)	
RELATIONSHIP (1:1 / 1:n / n:m)	
ATTRIBUTE (total/partial)	
IDENTIFIER (internal/external)	
REPEATING ATTRIBUTE	
AGGREGATE	
GENERALIZATION HIERARCHY with underlying attribute: a	
SUBSET HIERARCHY	
COMPLETE HIERARCHY	

Table 1

Abstract Knowledge

General constraints, different from those for which a special syntactic form is provided in the data models, can be stated in natural language in EERM and in the conceptual language Galileo in SDM /1/.

Procedural Knowledge

Several forms are provided to define special operations for manipulating objects in semantically meaningful ways, and to specify access requirements.

In the requirements collection and analysis phase the procedural knowledge is described by Operations Glossary. For each operation (transaction) the following aspects are specified: a natural language description; operation type (read, insert, delete); execution type (on-line, batch); frequency of execution; data involved and their role (input, output, visited) /13/.

In the conceptual design phase, with the approach based on EERM, operations can be defined at three levels of abstraction: conceptual, navigational and executable. The goal of a conceptual description is to specify the names and types of the data needed by an operation to ensure that the operation view is consistent with the conceptual schema. The goal of a navigational description is to specify the data needed by an operation, together with: a) the sequence in which tha data are accessed; b) the type of usage (read, insert, delete); c) the quantitative information on data usage in a period of interest (the number of times that an access step is executed, and an estimate of the number of records accessed by each such step). Finally, the goal of an executable description is to give an operational specification. For each level both a textual and a diagrammatic representation is provided /4/.

With the approach based on SDM, the procedural knowledge is described for three different purposes using the conceptual language Galileo. First, to specify derived information, such as derived attributes and derived classes. Second, to define domain-dependent operations to manipulate objects in a semantically meaningful way. The role of these operations is similar to those associated with abstract data types. Third, to define procedures guaranteed to leave the database in a consistent state (transactions), providing moreover facilities for raising and handling exceptions /1/. A version of Galileo has been defined to specify operational requirements at different levels of abstraction in a non-executable language including a construct to specify quantitative data during the requirements collection and analysis phase /2/.

Dynamics

In the requirements collection and analysis phase, dynamics is described by Events Glossary, which deals with the following aspects: precedence relationships among events; data and operations involved. For each event an Event Specification Form is filled to give information on the related conditions and operations /13/.

In the conceptual design phase, dynamics is described by means of Petri Nets or by processes /2,9/. Petri Nets are used with the

Condition-Event interpretation and they have been extended with definitional capabilities to deal with message passing among the functions to be automated. Processes are modeled in Galileo, which has been extended with a construct similar to 'scripts' of Taxis /2,3/.

3. THE DATAID METHODOLOGY

The aim of a database design methodology is to transform a user-oriented linguistic representation of the information needs of an organization into a DBMS-oriented description. This process is performed through several phases.

In the requirements collection and analysis phase, user requirements are translated into a requirements schema which is: user independent (that is, requirements are normalized according to established standards); and conceptual model independent (that is, no choice is made at this level regarding the structures of a model used to represent the information of interest). In the conceptual design phase, the requirements schema is formalized into a conceptual schema which is DBMS independent (in that, no choice is made at this level regarding the implementation structures). Finally, in the logical and physical design phases, the result is a DBMS dependent schema, that is, it is a DBMS processable schema.

The DATAID-1 methodology covers all these phases as illustrated in the following /10/. It assumes that the analysis of the organization, of its information flows, and the cost-benefit analysis have already been performed before the actual start of the database design process.

Inputs of the requirements collection and analysis phase are informal and heterogeneous descriptions of the observed system. Two classes of descriptions are considered: natural language sentences (derived from pre-existing documents, results of interviews or meetings, paper files); traditional/DBMS files. A recent extension of DATAID-1 deals with forms analysis as well /5/.

Output of the phase is a collection of glossaries describing data, operations, and events. To fill in the glossaries, natural language descriptions are filtered and rewritten in a restricted language; revised requirements are then classified into different sentence types (data, operations and events sentences). Intra-glossary and inter-glossaries checks concerning completeness and consistency are performed to prevent transmission and amplification of errors.

In the conceptual design phase a conceptual view is built for each environment of the organization (view modeling); these views are then integrated to form a global conceptual description of the database (view integration). The conceptual schema of an environment is the formalized representation of both the static and dynamic requiremants. As regards the conceptual model, we have adopted the EERM for data, constraints and operations modeling, and Petri Nets for functions modeling.

The design of local views is based on operations modeling. For each operation, an operation schema is built which describes the required data. The data schema is built progressively: when an operation schema is completed, data structures which have been introduced are aggregated with the previous partial data schema. This process is iterated for all the operations and, eventually, it produces the data schema. To start the aggregation process, we consider two alternative cases. In one case we use a skeleton schema: this happens when in the organization data are found naturally structured into entities, attributes and relationships. In the other case, an initial operation schema is built for the most relevant operation.

The organization functions are analyzed in terms of events, and are represented by Petri Nets graphs. These graphs show the causal dependencies/independencies between events (by means of structures of sequence, conflict and concurrency) and represent the way in which the functions to be automated must be executed.

In the views integration process, strategies (based on conflict analysis, merging, enrichment, and restructuring) are provided to deal with the problem deriving from the fact that objects (and their properties) which are common to different views have not been modeled in the same way (i.e. by means of the same name, classification structures, and integrity constraints structures) in the different schemata. After the integrated data schema has been constructed, integrated operation schemata are produced, and the events schemata are coordinated through communication links which represent message exchanges between functions.

The process of logical design maps from the conceptual schema to a logical schema of the DBMS chosen for the implementation. Two classes of DBMS are considered in the present version of the methodology: relational DBMSs and network (Codasyl-like) DBMSs.

The transformation process is based on two fundamental tasks:

. Simplification of the global conceptual schema: data structures not directly translatable into the logical model (like generalization hierarchies and multiple relationships) are converted into simpler ones. This task produces a simplified schema which contains only entities and binary relationships.

. Refinement of the simplified schema: a set of transformations on the simplified schema is applied (typically, partitioning of the entities and replication of attributes); performance measures are used in order to select, among different alternatives, the solution which optimizes the execution of the most important operations.

The aim of the physical design phase is to provide database designers with a complete framework of design decisions and to guide them through the design process to select the best physical design.

The physical design decisions for Codasyl-like databases are subdivided into three broad decision areas:

. Access path support decisions: implementation strategies for entry point records (LOCATION MODE clause options) and sets (SET MODE clause options) are considered.
. Placement strategy decisions: member records of a set are dispersed throughout the database area or clustered so that neighbouring member records tend to be stored in the vicinity.
. Storage allocation decisions: database areas for storage of records, indexes and pointer array tables are selected; each area is subdivided into a number of pages and the page length is fixed.

The methodology is based on the evaluation of all possible record and set implementation strategies from the global processing point of view. This is accomplished in the following tasks:

. Creation of record usage trees: accesses to a record are globally described as a tree where the leaf nodes are the different types of operations performed on the object record.

. Storage allocation: heuristic rules are used for calculating record length, record allocation in areas, area and page sizes, and other physical parameters.

. Evaluation of implementation strategies: starting from the relative costs and frequencies of the operations, implementation strategies are evaluated.

The main goal of the physical design for relational DBMSs is the selection of the secondary indexes of the relations of the schema; the choice of primary indexes (or keys) has already been made in the logical design phase.

The method is based on the following assumptions, that are quite realistic in the case of small/medium size DBMSs:

a) The indexes are structured as B^+-trees.
b) At most one index per relation can be used to access tuples in executing an operation.
c) Joins are performed according to the nested loops method.
d) The primary key cannot be updated and an index cannot be used to access tuples if it is currently modified.
e) The criteria used by the optimizer to evaluate execution costs are known.

The physical design is then consistent with the choices made by the optimizer and consists of four tasks:

. Specification of database statistics: these data are obtained transforming the results of the logical design phase.

. Cost evaluation: it provides a matrix giving the costs of all operations, each being executed when only one of all possible indexes is built on the relations.

. Index comparison: indexes are compared and the less efficient are eliminated.

. Generation of an efficient set of indexes: the final relational schema, in which all secondary indexes are determined, is constructed.

Classification of Methodologies and the DATAID Role

In the following we consider some classification criteria for methodologies and specify the role of the DATAID-1 methodology with respect to them. Methodologies can be classified according to several features.

A first classification distinguishes data-oriented methodologies which focus on properties of data (to enhance the stability of the design with respect to changes of the application), and function-