科技资料

Entity-Relationship Approach to Database Design & Querying

ENTITY-RELATIONSHIP APPROACH TO DATABASE DESIGN AND QUERYING

Proceedings of the Eighth International Conference on
Entity-Relationship Approach
Toronto, Canada, 18-20 October, 1989

edited by

Frederick H. LOCHOVSKY
Department of Computer Science
University of Toronto
Toronto, Ontario
Canada

1990

NORTH-HOLLAND • AMSTERDAM • NEW YORK • OXFORD • TOKYO

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

Distributors for the United States and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY INC.
655, Avenue of the Americas
New York, N.Y. 10010, U.S.A.

Library of Congress Cataloging-in-Publication Data

International Conference on Entity-Relationship Approach (8th : 1989 : Toronto, Ont.)
Entity-relationship approach to database design and querying : proceedings of the Eighth International Conference on Entity-Relationship Approach, Toronto, Canada, 18-20 October 1989 / edited by Frederick H. Lochovsky.
p. cm.
Includes bibliographical references.
ISBN 0-444-88716-4
1. Relational data bases--Congresses. 2. Data base design--Congresses. 3. Data base management--Congresses. I. Lochovsky, Frederick H. II. Title.
QA76.9.D26I57 1989
005.75'6--dc20 90-35666
CIP

ISBN: 0 444 88716 4

PRINTED IN THE NETHERLANDS

Preface

The Entity-Relationship conferences are organized for the purpose of bringing together researchers and practitioners whose focus is the use of the Entity-Relationship model in their work. The Eighth International Conference on Entity-Relationship Approach was held in Toronto, Canada October 18-20, 1989. Over 200 attendees from all parts of the world participated in the program which consisted of 9 paper sessions, 4 tutorials, and 4 panels.

The conference reflected the trends in recent years of extending the modeling power of the ER model and of incorporating knowledge-based techniques into design tools for and implementations of ER-based systems. The keynote address by William Kent of *Hewlett-Packard Laboratories* outlined the increasing trend to object-orientation in data models and user interfaces. The four tutorials covered the topics:

An Introduction to the Entity-Relationship Model (by M.E. Modell),

Visual Query Languages (by T. Catarci and C. Batini),

Artificial Intelligence and Databases (by Y. Ioannidis), and

Distributed Database Design (by T.J. Teorey).

The four panels addressed issues of:

User Experience with ER Modeling,

Toward More Power for Database Systems, Models, and Users—What Should We Be Doing?,

Do We Really Need Object-Oriented Data Models, and

Beyond SQL: Query Languages for the 90's.

This volume contains 23 papers which were presented at the Eighth ER Conference. Papers of high quality were solicited on both principles and pragmatics of using the entity-relationship approach in research and business. The papers in this volume deal with two broad topics: database design and database querying. The database design papers cover areas dealing with extending the ER data model, design methodologies, database design tools, and applications of knowledge-based techniques to ER database design tools and systems. The database querying papers cover areas dealing with schema and query translation, ER query languages, and graphical query interfaces.

Thanks are due to many people whose efforts made the conference a success. The authors submitted high-quality papers and produced final versions on schedule. The program committee members and referees reviewed the papers efficiently (each submitted paper was reviewed by three referees) and prepared suggestions and constructive criticisms for improving the quality of the papers. The Conference Chairmen, James P. Fry and Carlo Batini, worked hard to ensure that the conference was well publicized and adequate resources were available for producing an excellent program. James P. Fry also handled many of the non-program details of the conference including registration and, along with his assistant, Kim Mastan, did an excellent job making sure that everything ran smoothly. Dr. Peter P. Chen gave his support by working closely with North-Holland Publishing Company to have the proceedings published and handling various

details involved in running a conference. Mr. Lou F. Melli, Prof. Sham B. Navathe, and Prof. Alberto O. Mendelzon organized panel sessions. Sandy Choi, Mariano Consens, Isabel Cruz, John DiMarco, Chris Knight, Jeff Lee, Carlos Mendioroz, Dimitris Plexousakis, and Thodoros Topaloglou helped with registration and various errands that inevitably arise during the course of a conference.

Frederick H. Lochovsky
Editor

The 8th International Conference on

ENTITY-RELATIONSHIP APPROACH

Toronto, Canada **18-20 October, 1989**

US Conference Chairman

James P. Fry
University of Michigan

European Conference Chairman

Carlo Batini
Università di Roma

Program Committee Chairman

Frederick H. Lochovsky
University of Toronto

Vendor Exhibits Coordinator

Beverly Kahn
Suffolk University

Steering Committee Chairman

Peter P. Chen
Louisiana State University

Steering Committee Vice-Chairman

Salvatore T. March
University of Minnesota

Program Committee

Don Batory	The U. Texas at Austin (USA)
Joachim Biskup	Hochschule Hildesheim (West Germany)
Lynn Brady	Macquarie U. (Australia)
Marco A. Casanova	IBM Brazil (Brazil)
Ignacio R. Casas	U. Catolica de Chile (Chile)
Edward P.F. Chan	U. Waterloo (Canada)
Francis Y.L. Chin	U. Hong Kong (Hong Kong)
Robert C. Goldstein	U. British Columbia (Canada)
Thanasis Hadzilacos	U. Patras (Greece)
Yannis Ioannidis	U. Wisconsin Madison (USA)
Alwyn H. Jones	The City U. London (England)
Gerti Kappel	U. de Genève (Switzerland)
Peter Lyngbaek	Hewlett-Packard Laboratories (USA)
Yoshifumi Masunaga	U. Library and Inf. Sc. (Japan)
Dennis McLeod	U. Southern California (USA)
Lou Melli	Acumen Inf. Systems Inc. (Canada)
Alberto O. Mendelzon	U. Toronto (Canada)
Tim H. Merrett	McGill U. (Canada)
Barbara Pernici	Politecnico di Milano (Italy)
Ferdi Put	Katholieke U. Leuven (Belgium)
Fausto Rabitti	IEI-CNR Pisa (Italy)
Colette Rolland	U. Paris 1 Pantheon-Sorbonne (France)
Edward Sciore	Boston U. (USA)
Jacob Slonim	IBM Canada Ltd. (Canada)
Lars Soderlund	Neotech AB (Sweden)
Henk G. Sol	Delft U. Technology (The Netherlands)
Arne Solvberg	U. Trondheim (Norway)
Paul G. Sorenson	U. Alberta (Canada)
Toby J. Teorey	U. Michigan (USA)
A. Min Tjoa	U. (Austria)
Peter T. Wood	U. Cape Town (South Africa)

ER
APPROACH

Referees

All papers were reviewed by the Program Committee. In addition, invaluable help was provided by the referees listed below.

Arapis, C.

Bouggemann, H.H.
Braga, A.P.
Breiteneder, C.J.
Byeon, K.J.

Casais, E.
Chen, A.
Conveut, B.

Dur, R.C.J.

Fang, D.
Frenkel, G.
Furtado, A.L.

Li, L.

Navon, J.
Nussbaum, M.

Pissinou, N.

Ter Bekke, J.H.
Tucherman, L.

Verhoef, T.F.

Zezula, P.

Contents

1. PERSPECTIVE AND FUTURE DIRECTIONS

Entity-Relationship Approach to Database Design and Querying
F.H. Lochovsky (Editor)
Elsevier Science Publishers B.V. (North-Holland)

THE LEADING EDGE OF DATABASE TECHNOLOGY

William Kent

Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, California 94304 USA

1. PURPOSE

The nature of information modeling is shaped by the purpose at hand. General models of knowledge and cognition are most diverse and debatable, having the least defined criteria for correctness or adequacy. Models and methodologies in support of current and emerging information processing technologies are more tractable, being governed by the forms and capabilities of such technologies. These approaches can at least by judged by pragmatic measures of usefulness, if not correctness.

Information processing technology is evolving on several fronts, such as artificial intelligence, expert systems, robotics, and object oriented programming and database. Object orientation appears to be the next technology to mature. The time for its impact on information modeling methodologies is now.

Object orientation is far more than just another data model. It represents a substantial paradigm shift, calling for new ways of thinking about data and application development.

2. OBJECT ORIENTATION: THE NEXT STAGE OF EVOLUTION

We are at the confluence of two streams in information processing, embodied in those two words themselves: information and processing.

The dichotomy is reflected in a long history of parallel concepts: program and data, process and structure, procedural and declarative specifications, application development and database design, functional decomposition and data analysis.

Object orientation was born in the programming community, rediscovering much that was already known to the data community in terms of entity-relationship models and generalization concepts. A major innovation was the incorporation of procedural methods into the object construct.

Much of the confluence has been foreshadowed in programming disciplines. Lisp integrates the program and data space, treating procedures as data objects. The central principle of abstract data types is that data constructs are described in terms of their behavior rather than their structure.

On the data side, functional and process-oriented models have been proposed as data modeling approaches. The object-oriented class/type construct is an amalgam of entity types from data modeling and data types from programming languages.

Object orientation unifies these two streams into a single discipline, blending data structure and program behavior. These dual aspects bear a striking resemblance to

other dualisms of modern thought: matter and energy, particle and wave. All of these reflect a fusion of statics and dynamics.

The dualism requires a shift in our fundamental modes of thinking. The principle of abstract data types becomes central: the essence of an object is described in terms of its behavior. Structure is more a matter of implementation than semantics. We're going to grow beyond the spatial metaphors of structure.

The new way to think about data models is not as spatially laid out structures, but in terms of behavior. Views of data in spatial layouts are still invaluable as communication devices, both for people and for applications. But at bottom we need to recognize that the existence of a structure is manifest only by its behavior under various operations. What really goes on inside a machine bears little resemblance to the pictures in our imaginations.

The next stage in the evolution of information processing is an object-oriented unification of the programming language and data manipulation disciplines. It is as different from relational concepts as relational was from the navigational data languages of the hierarchical and network models.

3. SHIFTING BOUNDARIES: NEW ROLES AND INTERFACES

This unified duality of form and function calls for the integration of data and program development, organized around new roles and interfaces.

Although object orientation is very much an evolving technology, one of its central principles is data abstraction. Applications process data, not by manipulating data structures, but by applying *operations* (or *messages*) to objects. Those operations are executed by separately defined *methods* which isolate the application from the structures in which the data is implemented. The operations are defined in terms of the semantic entities of the enterprise, e.g., installing chips on boards, or printing documents, and not in terms of the constructs in a data model such as records, rows, or columns.

Thus object orientation is not just another structural form in the tradition of hierarchies, networks, and relations. It's a paradigm shift, a radical wrenching of the way we think about things.

This is not the first time. It happened when programmers stopped thinking in terms of data on devices, and had to think more abstractly in terms of data structures. It happened when programmers had to stop thinking about how to navigate around in data structures, and could simply describe what they wanted from the data structures.

Now we have to stop thinking about what a data object looks like, and think instead about how it acts.

The boundary between programs and the persistent data they operate on is shifting — again.

In the beginning, applications directly operated input/output devices to read and write data. Very quickly there evolved layers of interfaces shielding programs from these devices, such as device drivers and access methods. Soon programs became less and less conscious of which device the data was on, or even what kind of device, or whether it was on a storage device at all.

Databases provide more sophisticated data services, such as recovery and concurrency control, and increasingly technology-independent data structures in which to manage the data: hierarchical, network, relational, and even the various entity-relationship and semantic models.

But a fairly clear boundary has remained between programs and data:

applications	application code	program
data support	data structures	structure

Programs are outside the database, data is inside, and data structures provide the interface by which programs manage data. Even though queries are, in a sense, procedures executed in the database on behalf of the application, the interface seemed clear: application code operated on data structures.

Now the boundary is blurring. In object orientation, methods are programs which provide access to data, and those programs appear to be part of the data. Database technology, which now constitutes a single interface between programs and data, is splitting into two layers:

applications	application code	
data support	data operations	program
	data structures	structure

Process and data blend into a new hybrid in the middle ground. Data operations take on both aspects. They don't necessarily have to be defined procedurally, like programs. Sometimes it's enough to define static mappings to data structures.

The shifting boundary means that more application code will shift into the database.

There's a corresponding upward shift in database management. Database developers will become responsible for providing methods as the data operations by which users access data. How those operations map to data structures becomes the data developer's business.

The division of responsibility used to be this: data developers exposed data structures to applications, but privately worked out how those structures mapped to storage and device facilities. Now the division of responsibility is moving up a level: data developers expose data operations to applications, but privately work out how those operations map to data structures.

This implies some organizational shifts as well, with more programming associated with database development. *Method writer* is likely to emerge as a new role in database management, distinct from application programmers and system programmers. New technology is emerging for method writers. Methods can be written to map to conventional data structures, as well as to new structural models being developed for object oriented databases. While such structures provide greater efficiency for complex data, they are not exposed at application interfaces.

Data independence, the commitment to stability, is at a higher level. Data administrators have a much greater degree of freedom in what they can reorganize and optimize without impacting application programs.

Data operations constitute a new middle ground in this technology, an intersection of programming and data. In one sense, they were always there, but wired into the system in the form of operations on the data structures, e.g., the relational operators. They are evolving into something provided by data designers, dealing with data objects appropriate to the applications rather than system objects built into the database.

Data operations are hybrids, being programs that are aware of data structures. They relate to data in both styles. A data operation manipulates the internal structure of its own kind of object. But, if it needs other data, it is only allowed to invoke the data operations of other objects, and not play with their internal structure. This isolation extends the protection of data independence even further. When the internals of a particular data object change, it is only the data operations for that object which have to be redefined. Not only are application programs shielded from this change, but so are the data operations for other objects.

This all sounds a lot like subroutine libraries. What's the difference? Data operations are centrally owned and managed, part of the database. Data operations are not optional; they are an enforced discipline. Applications can't get around them and use the data structures directly if they feel like it. The data dictionary for application programmers should only show them the public interfaces supported for data. Also, data operations don't have to be defined procedurally like program subroutines.

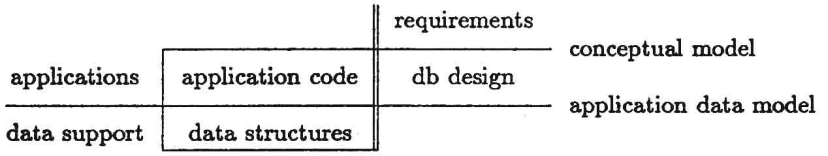
These data operations will in fact become units of reusable code.

The middle ground represents a mapping from semantics at the upper interface to implementation at the lower interface. The dream of closing the semantic gap is itself being realized: the conceptual schema is the application interface. Applications are expressed in terms of operations on objects, which is just entities and relationships in modern dress. The entity concept is enriched with such enhancements as subtypes. The messages which constitute the data operations subsume relationships, attributes, and procedures.

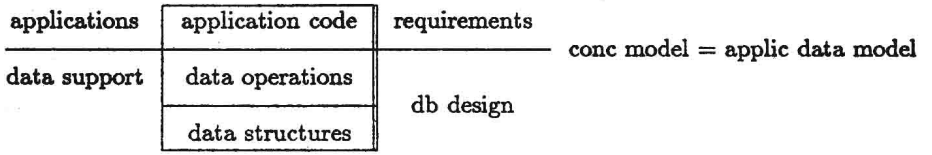
Incorporating structure and process into data operations should force integration of data and application development methodologies. There is a heightened need for improved formal specifications of behavior. Object oriented technology is still rather primitive in this respect. While the goal is to separate behavior from implementation, so-called specifications of behavior currently only constrain the types of the results returned by operations, not the correctness of values.

The nature of application execution will evolve, with more and more shifting from the programming system's execution space to the database's execution space. The two spaces might even merge. Responsibilities for managing storage space may shift. Allocating, managing, reclaiming space for objects may be more in the province of the object manager than the application program. Program systems and environments may focus more on specifying algorithms, less on data formats and structures, heap management, garbage collection, etc. New assumptions about stability: persistence could become the default. Shifting more and more procedurality into the database will have major impact on who does optimization, and how.

The focus of various stages of development will shift. Currently the conceptual schema is seen as just a formalization of requirements, not directly usable by applications. The database design process is required to turn these into something that applications can use:



In the new world, the conceptual schema directly defines interfaces to be used by applications. Database design, instead of providing stable data structures for direct use by applications, will concentrate on tuning and re-tuning physical designs to meet shifting performance needs, with method definitions revised as needed to isolate applications from such change:



4. CONCLUSIONS

The emerging technology of object orientation will have profound impact on the nature of information modeling and application development methodologies. Definition of process and structure will necessarily be integrated, oriented around a new execution environment in which applications perceive data behaviorally. Data operations will emerge as a new mediator between applications and data, serving as a library of reusable code owned by the database to provide a higher level of data independence for applications. The conceptual schema itself will serve as a directly usable application interface.