

Lecture Notes in Computer Science

759

Nabil R. Adam Bharat K. Bhargava (Eds.)

Advanced Database Systems



Springer-Verlag

A244 Nabil R. Adam Bharat K. Bhargava (Eds.)

Advanced Database Systems

Springer-Verlag

Berlin Heidelberg New York

London Paris Tokyo

Hong Kong Barcelona

Budapest

Lecture Notes in Computer Science

759

Edited by G. Goos and J. Hartmanis

Advisory Board: W. Brauer D. Gries J. Stoer



Contents

I	Object-Oriented Databases	1
1	COMPOSE: A System for Composite Specification and Detection	3
by N. Gehani, H.V. Jagadish, O. Shmueli		
1.1	Introduction	3
1.2	Event Expressions	4
1.2.1	Basic Operators	5
1.2.2	Additional Operators	5
1.2.3	Regular Expressions	6
1.3	Examples	7
1.3.1	Simple Examples	7
1.3.2	Discount Rate Cut	7
1.3.3	Attributes and Masks	8
1.3.4	Parameters	9
1.3.5	Correlation Variables	10
1.4	Composite Event Detection	10
1.4.1	Design Decisions	11
1.4.2	Masks	11
1.4.3	Generic Automaton For Implementing Events With Parameters	12
1.5	Compose System	12
1.6	Examples of Automata Generated by Compose	13
1.7	Conclusion	14
2	Access Controls in OO Database Systems – Some Approaches and Issues	17
by E. Bertino, S. Jajodia, P. Samarati		
2.1	Introduction	17
2.2	Object-oriented Data Model	19
2.3	Mandatory Access Control	21
2.4	Discretionary Access Control	25
2.4.1	The ORION Authorization Model	25
2.4.2	Content-dependent Authorizations	34
2.4.3	Accessing Objects Through Methods	35
2.5	Research Issues in Mandatory Access Control	38
2.5.1	Modeling Multilevel Entities as Single-level Objects	38

2.5.2	Object Updates and Secure Garbage Collection Mechanisms	40
2.5.3	Polyinstantiation	41
2.5.4	Comparison With Relevant Work	42
2.6	Research Issues in Discretionary Access Control	43
2.7	Conclusion	44
3	The Decomposition Property of Non-Deterministic Databases	45
by K. Vadaparty, S. Naqvi		
3.1	Introduction	45
3.2	Basic Notions	49
3.2.1	Entailment, and Data Complexity	49
3.2.2	Choices and Data Complexity	50
3.3	Size and Density of a Witness	52
3.3.1	Size of a Witness	53
3.3.2	Density of Witnesses	56
3.4	Modulewise Evaluation	56
3.5	Future Extensions	63
4	The Architectures of an Object Base Environment for Simulation	65
by P.C.-Y. Sheu, L.J. Peterson		
4.1	Introduction	65
4.2	Related Work	66
4.3	Object Representation	67
4.3.1	Complex Objects	67
4.3.2	Active Objects and Models	68
4.4	Management of Active Objects	71
4.4.1	State Space and Criteria	71
4.4.2	Adding and Removing A State	71
4.4.3	Adding and Removing A Rule	72
4.4.4	Adding and Removing An Attribute	73
4.4.5	Adding and Removing A Method or A Class	73
4.5	Simulation	73
4.5.1	Rule Processing	73
4.5.2	Logic of The Simulator	78
4.5.3	Parallel Processing	79
4.5.4	Example	80
4.6	Object-Oriented Evaluation of Rule Networks	81
4.6.1	Structures of Extensional Databases and Query Networks	81
4.6.2	Object-Oriented Rule Evaluation	83
4.7	Conclusion	85

5	Transition from a Relation to Object Model Implementation	87
	by B. Bhargava, Y. Jiang, J. Srinivasan, P. Dewan	
5.1	Modeling Complex Data	87
5.2	Survey of Extended Relational Systems	88
5.3	O-Raid System Design and Implementation	90
5.3.1	Expand Query Language	90
5.3.2	Extend Data Definition Facility	91
5.3.3	Data Manipulation Language (DML)	97
5.4	Performance Studies	99
5.5	Research Issues	102
6	An Object-Oriented Knowledge Model for KBMS-Supported Evolutionary Prototyping of Software Systems	105
	by S.Y.W. Su, Y. Shyy	
6.1	Introduction	105
6.1.1	Motivation	105
6.1.2	Related Works	106
6.2	Knowledge Model Overview	107
6.2.1	Classes	107
6.2.2	Objects and Instances	108
6.3	Structural Abstraction	108
6.3.1	Structural Association Definitions	108
6.3.2	Encapsulation and Inheritance	109
6.3.3	Extensible Kernel Model	109
6.3.4	Structural Association Patterns	110
6.4	Behavioral Abstraction	111
6.4.1	Method Model and Control Associations	112
6.4.2	Method_model Object and Evolutionary Prototyping	115
6.4.3	Rule Definition	116
6.5	Conclusions	117
7	Applying OOAD in the Design and Implementation of an Intelligent Geographic Information System	127
	by R. Subramanian, N.R. Adam	
7.1	Introduction	127
7.2	Modeling & Query Processing	128
7.3	Spatial Data Modeling	130
7.3.1	The Design Methodology	130
7.4	The Responsibility-Driven Approach	131
7.5	Developing the Data Model	132
7.5.1	The Exploratory Phase	132
7.5.2	The Analysis Phase	139
7.6	Implementation	144
7.7	Conclusion	145

II Temporal/Historical Database Systems 151

8 Indexical Databases 153

by J. Clifford

- 8.1 Motivation 153
- 8.2 The Indexical Database Model 155
 - 8.2.1 The Structures 155
 - 8.2.2 Discussion of the Structures 156
 - 8.2.3 An Indexical Example: The Watergate Database 158
 - 8.2.4 The Operations 159
 - 8.2.5 Partial Functions 168
- 8.3 Instances of the Indexical Database Model 172
 - 8.3.1 HRDM 172
 - 8.3.2 Bitemporal Database Models 173
- 8.4 Summary and Conclusions 173

9 A Temporal Query Language for a Conceptual Model 175

by R. Elmasri, V. Kouramajian

- 9.1 Introduction 175
- 9.2 Representing Time 176
- 9.3 The Temporal Data Model 176
 - 9.3.1 Conceptual Objects: Entities 176
 - 9.3.2 Temporal Objects: Roles 177
 - 9.3.3 Temporal Constraints among Roles 178
 - 9.3.4 Non-Temporal Attributes 178
 - 9.3.5 Temporal Attributes 178
 - 9.3.6 Classes and Superclass/Subclass Relationships 179
 - 9.3.7 Conceptual Relationships 179
 - 9.3.8 Temporal Relationships 180
 - 9.3.9 Temporal Constraints among Relationships 180
 - 9.3.10 An Example 180
- 9.4 Temporal Query Language Constructs 181
- 9.5 The Temporal Query Language 183
 - 9.5.1 Temporal Projection 184
 - 9.5.2 Temporal Selection 187
 - 9.5.3 Temporal Version Restriction Operators 188
 - 9.5.4 Temporal Scope Operators 190
- 9.6 Conclusions 190

10 A Data Model for Time-Series Analysis 191

by A. Segev, R. Chandra

- 10.1 Introduction 191
- 10.2 Main Features of The Data Model 192
 - 10.2.1 Relevant Research 193
- 10.3 Vector Based Data Model 194

10.4 Concepts	198
10.5 Rules	203
10.6 Calendar	204
10.7 Temporal Query Language	205
10.8 Special Operators for Time-Series Database	206
10.9 Handling of Missing Values	208
10.10 User Environment	208
10.11 Conclusion	209
11 A Relational Model and SQL-like Query Language for Spatial Databases	213
by S.K. Gadia, V. Chopra	
11.1 Introduction	213
11.1.1 Related Works	213
11.1.2 Our Concept of A Spatial Region	214
11.1.3 Weak Data Typing	215
11.1.4 Uniformity of Attribute Values	215
11.1.5 Experience From Temporal Databases	215
11.2 Our Model	216
11.2.1 Spatial Regions	216
11.2.2 Attribute Values	216
11.2.3 Value Navigation	217
11.2.4 Spatial Tuples	217
11.2.5 Spatial Relations	218
11.2.6 Weak Equality and Restructuring	219
11.3 Querying in The Model	219
11.3.1 Spatial Expressions	219
11.3.2 Boolean Expressions	220
11.3.3 Relational Expressions	220
11.3.4 Examples	221
11.4 Seamlessness of SpaSQL	222
11.5 Algebraic Nature of SpaSQL	224
11.6 Conclusion	224
III Query Processing in Database Systems	227
12 Parallel Query Processing	229
by P.S. Yu, M.-S. Chen, J.L. Wolf, J. Turek	
12.1 Introduction	229
12.2 Preliminaries	230
12.3 Issues	231
12.3.1 Intra-operator Parallelism	232
12.3.2 Inter-operator Parallelism	232
12.3.3 Inter-query Parallelism	233
12.3.4 Remarks	234
12.4 System Architectures	234

12.5 Data Skew and Intra-operator Parallelism	235
12.5.1 Conventional Algorithm	238
12.5.2 Dynamic Algorithms	239
12.5.3 Sophisticated Algorithms	240
12.6 Complex Multi-join Query	242
12.6.1 Join Methods without Pipelining	243
12.6.2 Join Methods with Pipelining	248
12.7 Scheduling Multiple Complex Queries	252
12.7.1 A Hierarchical Approach to Inter-Query Parallelism . . .	253
12.7.2 Scheduling Independent Tasks	255
12.8 Summary	258
13 Towards Flexible Distributed Information Retrieval	259
by D.W. Flater, Y. Yesha	
13.1 Introduction	259
13.2 Information Retrieval Techniques	260
13.2.1 Tag-Based Retrieval	260
13.2.2 Partial Content-Based Retrieval	261
13.2.3 Full Content-Based Retrieval	261
13.3 Thesauri and Dictionaries	261
13.3.1 Thesauri	262
13.3.2 Dictionaries	263
13.4 Fuzzy Retrieval	263
13.4.1 Partial Content-Based Methods	264
13.5 Distributed Approaches to Information Retrieval	265
13.5.1 Current Research Issues	266
13.6 Architecture	267
13.6.1 Preliminaries	267
13.6.2 Query Routing	268
13.6.3 Cooperative Caching	268
13.6.4 Simulation Results	269
13.6.5 Implementation	272
13.7 Concluding Remarks	276
14 Efficient Parallel Recovery in Replicated Databases	277
by R. Tewari	
14.1 Introduction	277
14.2 Consistency Control Algorithm	278
14.3 Parallel Merge Protocol	279
14.4 Extension of the Parallel Merge Protocol	285
14.5 Incorporating Parallelism in the Merge Protocol	285
14.6 Performance Analysis of the Parallel Merge Algorithm	286
14.7 Conclusion	287

15 Document Allocation in Multiprocessor Information Retrieval Systems	289
by H.T. Siegelmann, O. Frieder	
15.1 Introduction	289
15.2 MDAP is NP-Complete	290
15.3 Related Efforts	292
15.3.1 Previous Approximations of the Mapping Problem	293
15.3.2 Related Information Retrieval Systems	293
15.4 A Genetic Algorithm for MDAP	294
15.5 Theoretical Foundations	298
15.6 Experimental Evaluation	301
15.7 Conclusions and Future Directions	309
 IV Heterogeneity/Interoperability/Open System Architectures/Multimedia Database Systems	 311
16 Amalgame: A Tool for Creating Interoperating, Persistent, Heterogeneous Components	313
by J.-C. Franchitti, R. King	
16.1 Introduction	313
16.1.1 The Persistent and Heterogeneous Applications	313
16.1.2 Goals and Novelty of the Amalgame System	316
16.2 Related Work	316
16.2.1 Interoperability Support	316
16.2.2 Architectural Representation Languages and Megaprogramming	317
16.2.3 Extensible Reusable Heterogeneous Frameworks	318
16.3 An Overview of Amalgame	318
16.3.1 A Motivating Example	318
16.3.2 The Designer's View of the Amalgame Toolkit	319
16.3.3 Practical Use of the Amalgame Toolkit	320
16.3.4 The Internal Architecture of the Amalgame Toolkit	325
16.4 An Arcadia Demonstration Scenario	330
16.4.1 High-level Description of the Arcadia Experiment	330
16.4.2 Implementation	331
16.4.3 Benefits of The Amalgame Approach	332
16.5 Future Directions	333
16.5.1 A Joint Arcadia and Prototech Demonstration Scenario	333
16.5.2 An Interoperability Experiment with Chiron	333
16.5.3 An International "Library" of Deployed "Wrapped" Persistent Applications	334
16.6 Conclusion	334
16.7 Acknowledgments	334

17 Correctness and Enforcement of Multidatabase Interdependencies	337
by G. Karabatis, M. Rusinkiewicz, A. Sheth	
17.1 Introduction	337
17.2 Background	338
17.2.1 Specification of Interdependent Data	338
17.2.2 Conceptual System Architecture	340
17.3 Correctness of Dependency Specifications	341
17.3.1 Dependency Graph	341
17.3.2 Correctness Requirements Involving Consistency Predicates	343
17.3.3 Correctness Requirements Involving Dependency Predicates	344
17.4 Polytransactions	345
17.5 Consistency of Interdependent Data	348
17.5.1 States of Interdependent Data Objects	348
17.5.2 Measures of Consistency	349
17.5.3 Updatability of Objects	351
17.6 Concurrent Execution of Polytransactions	353
17.6.1 Correctness of the Execution of a Single Polytransaction .	353
17.6.2 Conflicts in Polytransactions	354
17.6.3 Polytransactions with Temporal Constraints	356
17.7 Conclusion	357
18 FEMUS: A Federated Multilingual Database System	359
by M. Andersson, Y. Dupont, S. Spaccapietra, K. Yétongnon, M. Tresch, H. Ye	
18.1 Introduction	359
18.2 FEMUS	360
18.2.1 The ERC+ Approach	361
18.2.2 The COCOON Approach	362
18.3 The Mapping Process	365
18.3.1 Mapping an ERC+ Schema to COCOON	366
18.3.2 Mapping a COCOON Schema to ERC+	368
18.3.3 Operators Mapping	369
18.4 The Integration Process	371
18.4.1 Assertion-driven Integration	374
18.4.2 Integration Through Augmentation	375
18.5 Negotiation	376
18.5.1 Exchanging Metadata	376
18.5.2 Exchanging Data	377
18.6 Implementation Issues	377
18.7 Consistency Requirement	378
18.7.1 Detection of Relevant Updates	378
18.7.2 Differential Refresh	379
18.8 Conclusion and Future Research	380

19 Communication and Synchronization Issues in Distributed Multimedia Database Systems	381
by S. Browne	
19.1 Introduction	381
19.2 Characteristics and Requirements	383
19.3 Communication Approaches	385
19.4 Synchronization Approaches	388
19.5 Conclusions	395
20 Multimedia Database Systems	397
by A. Ghafoor, P.B. Berra	
20.1 Introduction	397
20.2 Characteristics of Multimedia Data	398
20.2.1 Text and Formatted Data	398
20.2.2 Audio and Music Data	398
20.2.3 Images and Pictures Data	398
20.2.4 Full-Motion Video Data	399
20.3 Notion of Time for Multimedia Data	399
20.3.1 The Temporal Synchronization Problem	399
20.3.2 Modeling Time	401
20.4 Conceptual Models for Multimedia Objects	402
20.4.1 Graphical Models	402
20.4.2 Petri-Net Models	403
20.4.3 Object-Oriented Models	404
20.4.4 Language Based Models	405
20.4.5 Temporal Abstraction Models	405
20.4.6 Database Models for Multimedia Synchronization	406
20.5 Some Multimedia Database Systems	407
20.5.1 Image Database	407
20.5.2 Audio Database	409
20.6 Challenges in Multimedia Database	410
20.7 Conclusion	411
Bibliography	413

Part I

Object-Oriented Databases

Chapter 1

COMPOSE: A System For Composite Specification And Detection

Narain Gehani^{*}, H. V. Jagadish[†], O. Shmueli[‡]

1.1 Introduction

An “event” is a happening of interest. Events can be simple such as, the stock price going above a certain price, the beginning of a transaction, the update of an object, or the temperature going above a specified limit. New events can also be formed as a combination of other events, for example, three successive discount rate cuts without an intervening increase, all withdrawals following a large deposit, and the temperature going above a specified limit and staying there for more than some time period. We call such events “composite events”.

We have developed a model for specifying composite events [216, 215]. We were motivated to explore the specification of composite events as part of an effort to design “trigger” facilities for the Ode object database [7, 217]. Triggers are the key facility that distinguishes active databases [138, 524, 40, 378, 556, 395] from passive databases. A trigger consists of an event-action pair. When an event is detected, the associated action is executed.

The use of triggers moves code from the application to the database. This simplifies application writing because the application now does not have to check for the conditions specified by the triggers. Triggers also eliminate duplicate code since the same conditions may have to be checked in multiple applications.

A trigger facility in which triggers fire on the occurrence of composite events is more powerful than one in which triggers fire on the occurrence of simple events because it allows users to write triggers that could not be easily expressed before. Composite event specification is useful for many application domains besides databases:

1. Financial Applications: Trades can be executed in response to an observed pattern of (trading) events in a stock market.

^{*}AT&T Bell Laboratories

[†]AT&T Bell Laboratories

[‡]AT&T Bell Laboratories and Technion -Israel Institute of Technology

2. Fraud Detection: Particular sequences of credit card purchases may point to fraudulent use.
3. Production Management: Particular sequences of defects could indicate difficulties that must be brought to the attention of a supervisor.

Composite events are specified as event expressions. Our basic notation for specifying composite events has the same expressive power as regular expressions. Thus the occurrence of a composite event is detected by a finite automaton that implements the event expression. Despite the equivalence of expressive power, our notation is specially suited for specifying composite events. For example, it allows for the easy specification of composite events whose components can overlap and allows uninteresting events to be screened out.

We extend our basic notation with “masks”, correlation variables, and parameters, thereby stepping beyond the domain of regular expressions. However, we can still implement event expressions that use these facilities by using automata augmented with “mask” events and by using “generic” automata. This allows us to use finite automata optimization techniques to generate efficient implementations for recognizing the occurrence of composite events.

We have built a prototype system, COMPOSE, for specifying and detecting composite events. A real-time stock trade feed is used to experiment with specifying and detecting stock market related events.

In this chapter, we describe how composite events are specified, illustrate composite event specification, give an overview of COMPOSE, and describe the construction of the finite automata.

1.2 Event Expressions

Primitive events are events that are known to or supported by the database system. Examples of some primitive events, in object-oriented databases [215], are object manipulation actions such as creation, deletion, and update or access by an object method (member function). Events can be specified to happen just prior to or just after the above actions. In addition, events can be associated with transactions and specified to happen immediately after a transaction begins, immediately before a transaction attempts to commit, immediately after a transaction commits, immediately before a transaction aborts, and immediately after a transaction aborts. Examples of other events are time events such as clock ticks, the passage of a day, an hour, a second, or some other time unit. Finally, stock trades and the raising or lowering of interest rates are examples of financial events, and company announcements are examples of news events.

Composite events are specified as event expressions. An *event expression* can be *NULL*, any primitive event *a*, or an expression formed using the basic operators $\wedge$, $!$ (not), *relative* and *relative+*.

Formally, event expressions are mappings from *histories* (sequences of primitive events) to histories:

$$E : \text{histories} \rightarrow \text{histories}$$

The result of applying an expression *E* to a history *h*, which is also a history, is denoted by $E[h]$.

Here are the semantics of some simple event expressions:

1. $E[\text{null}] = \text{null}$ for any event E , where null is the empty history.
2. $\text{NULL}[h] = \text{null}$.
3. $a[h]$, where a is a primitive event, is the maximal subset of h composed of all event occurrences of the symbol a .

1.2.1 Basic Operators

Let E and F denote event expressions and h, h_1, h_2 denote event histories. Here are the semantics of expressions formed using the basic operators:

1. $(E \wedge F)[h] = h_1 \cap h_2$ where $h_1 = E[h]$ and $h_2 = F[h]$.
2. $(!E)[h] = (h - E[h])$.
3. $\text{relative}(E, F)[h]$ are the event occurrences in h at which F is satisfied assuming that the history started immediately following *some* event occurrence in h at which E takes place.

Formally, $\text{relative}(E, F)[h]$ is defined as follows. Let $E^i[h]$ be the i^{th} event occurrences in $E[h]$; let h_i be obtained from h by deleting all event occurrences before $E^i[h]$. Then $\text{relative}(E, F)[h] = \bigcup_{i=1}^{E[h]} F[h_i]$.

4. $\text{relative} + (E)[h] = \bigcup_{i=1}^{\infty} \text{relative}^i(E)[h]$ where $\text{relative}^1(E) = E$ and $\text{relative}^i(E) = \text{relative}(\text{relative}^{i-1}(E), E)$.

1.2.2 Additional Operators

Besides the basic operators, we provide some additional operators that make composite events easier to specify. These operators do not add to the expressive power provided by the basic operators. Consequently, they can be defined in terms of the basic operators.

Let h denote a non-null history, and E, F , and E_i denote event expressions. The new operators are

1. $E \vee F = !(E \wedge !F)$.
2. *any* denotes the disjunction of all the primitive events.
3. $\text{prior}(E, F)$ specifies that an event F that takes place after an event E has taken place. E and F may overlap. Formally, $\text{prior}(E, F) = \text{relative}(E, \text{any}) \wedge F$.
4. $\text{prior}(E_1, \dots, E_m)$ specifies occurrences, in order, of the events $E_1, E_2, \dots, E_m$. $\text{prior}(E_1, \dots, E_m) = \text{prior}(\text{prior}(E_1, \dots, E_{m-1}), E_m)$.
5. $\text{sequence}(E_1, \dots, E_m)$ specifies immediately successive occurrences of the events $E_1, E_2, \dots, E_m$:
 - (a) $\text{sequence}(E_1, \dots, E_m) = \text{sequence}(\text{sequence}(E_1, \dots, E_{m-1}), E_m)$.
 - (b) $\text{sequence}(E_1, E_2) = \text{relative}(E_1, !(\text{relative}(\text{any}, \text{any}))) \wedge E_2$.

The first operand of the conjunction specifies the first event following event E_1 . The second operand specifies that the event specified by the complete event expression must satisfy E_2 .

6. *first* identifies the first event in a history.
 $\text{first} = !\text{relative}(\text{any}, \text{any})$.