

## Case 1. Introduction

## Case 2. Four hours to save the library

- 2.1 An emergency situation 1
- 2.2 First mission: recovering the data 2
- 2.3 The basic functions 2
- 2.4 The main window 3
- 2.5 Registering a new reader 4
- 2.6 Registering a new book 6
- 2.7 Borrowing a book 6
- 2.8 Returning a book 7
- 2.9 Updating the data of a reader 8
- 2.10 Querying reader and book data 8
- 2.11 Conclusion 12

## Case 3. The human factor

## Case 4. Interactive SQL interpreters

- 4.1 Introduction 2
- 4.2 The smallest interactive interpreter in the world! 2
- 4.3 A simple but realistic SQL interpreter 5
- 4.4 Error management 5
- 4.5 Adding information fields 9
- 4.6 Adding functions 9
- 4.7 SQL interpreters as training support 11
- 4.8 Semantic correctness of a query 12
- 4.9 An interactive SQL tutor 14
  - 4.9.1 The database 14
  - 4.9.2 The interface 15
  - 4.9.3 The tutor engine 17
  - 4.9.4 Improving the SQL tutor 18
- 4.10 The scripts 20

## Case 5. Schema-less databases - Part 1

- 5.1 Introduction 2
- 5.2 The Universal table model 2
  - 5.2.1 Generating the Universal table data from source tables 3
  - 5.2.2 From the Universal table back to the standard tables 3
  - 5.2.3 Application 4
  - 5.2.4 Origin of the Universal table model 6
  - 5.2.5 Comments 6
- 5.3 Column-oriented data model 7

- 5.3.1 Generating the Column-oriented data from the source tables 7
- 5.3.2 From the Column-oriented tables back to the source tables 8
- 5.3.3 Column-oriented model with abstract primary keys 9
- 5.3.4 About performance 11
- 5.3.5 Not all Column-oriented models are created equal 11
- 5.3.6 Comments 12

#### Case 6. Schema-less databases - Part 2

- 6.1 Introduction 1
- 6.2 The Key-Value model - Version 1 2
  - 6.2.1 Generating Key-Value data from the source tables 3
  - 6.2.2 The case of composite primary keys 5
  - 6.2.3 Rebuilding the source tables from their Key-Value expression 6
  - 6.2.4 About performance 7
  - 6.2.5 Application 9
- 6.3 The Key-Value model - Version 2 10
- 6.4 The Key-Value model - Version 3 11
  - 6.4.1 Application to Description Logic 13
- 6.5 Not all Key-value models are created equal 18
- 6.6 The scripts 19

#### Case 7. Schema-less databases - Part 3

- 7.1 Object view of Key-Value data 1
- 7.2 Objects with complex properties 3
  - 7.2.1 Multivalued properties 3
  - 7.2.2 Composite multivalued properties 5
- 7.3 Indexing objects 7
- 7.4 Comparison of the data models 9
- 7.5 For once, most Object data models are created equal 10
- 7.6 The scripts 11

#### Case 8. Schema-driven code generation

#### Case 9. Active databases

#### Case 10. Temporal databases - Part 1

- 10.1 Introduction 1
- 10.2 Representation of time 2
- 10.3 The concept of current state 4
- 10.4 History of an entity 5
  - 10.4.1 Temporal primary, unique and foreign keys 7
- 10.5 Transaction and valid times 9
- 10.6 Managing transaction time historical data 10

---

10.6.1	Current state of entities	12
10.6.2	Creation of an entity	14
10.6.3	Evolution of an entity	16
10.6.4	Deleting an entity	18
10.7	Managing valid time historical data	20
10.7.1	Current state of entities	21
10.7.2	Creation of an entity	21
10.7.3	Evolution of an entity	23
10.7.4	Deleting an entity	24
10.8	Alternative temporal database models	25
10.8.1	Table splitting	26
10.8.2	Attribute-based history	26
10.8.3	Event-based history	30
10.8.4	Document-oriented history	32
10.8.5	Normalized temporal database	33
10.9	The scripts	34
10.10	Some references	34
Case 11. Temporal databases - Part 2		
Case 12. Kings of France - Part 1		
12.1	Introduction	1
12.2	Data structures	3
12.3	Warming up: some easy queries	5
12.4	Rebuilding royal families	7
12.5	King succession	11
12.6	Seeking missing kings	11
12.7	Royal conflicts	13
Case 13. Kings of France - Part 2		
13.1	The ancestor/descendant relationships	1
13.2	Loop-based computing of the transitive closure	6
13.3	Counting descendants	6
13.4	Showing the descendants of a member	8
13.5	Graphical representation of king genealogy	10
13.6	Recovering the source data from their closure	11
13.7	Extracting the hierarchy of kings	13
13.8	The scripts	17
Case 14. Bis repetita non (semper) placent (data redundancy)		
Case 15. The book of which you are the hero		
15.1	Interactive adventure books and computer games	2

- 15.2 Choosing sample game books 2
- 15.3 Designing a text-based interactive adventure game 5
- 15.4 Story telling 8
- 15.5 Structure of a game 10
  - 15.5.1 Episodes, branches, paths and runs 10
  - 15.5.2 Starting and ending episodes 11
  - 15.5.3 Circuits 11
- 15.6 Analyzing a game 12
  - 15.6.1 Searching for game abnormalities 12
  - 15.6.2 Does the game include circuits? 13
- 15.7 Extracting circuits 16
- 15.8 Counting runs 20
- 15.9 Are all episodes reachable? 24
- 15.10 Do all episodes contribute to a solution? 26
- 15.11 Merging episodes 26
- 15.12 Toward a better game engine 27
- 15.13 The scripts 28
- Case 16. Directory management
  - 16.1 Of files and directories 2
  - 16.2 Examining directories 2
  - 16.3 Modeling the problem 3
  - 16.4 Data structures 5
  - 16.5 Loading a directory instance 8
    - 16.5.1 Creating a directory 8
    - 16.5.2 Creating the current instance 8
    - 16.5.3 Creating the nodes of the current instance 9
    - 16.5.4 Computing the path of a node 10
  - 16.6 Applications 12
  - 16.7 Application 1: general statistics 12
  - 16.8 Selecting a directory instance 14
  - 16.9 Application 2: Structure of a directory instance 15
  - 16.10 Application 3: Examining the contents of an instance 16
  - 16.11 Application 4: Duplicate nodes in a directory instance 17
    - 16.11.1 List of duplicate nodes in a directory instance 18
    - 16.11.2 Description of duplicate nodes in a directory instance 20
  - 16.12 Application 5: Comparing two directory instances 21
  - 16.13 Application 6: Searching for clone files 23
    - 16.13.1 The problem 24

---

16.13.2 Saved by (computer) science	24
16.13.3 Hashing files	25
16.13.4 Identifying clone candidates in a directory instance	26
16.13.5 Identifying clone candidates among two directory instances	27
16.14 A touch of optimization	29
16.15 Building the DIRECTORY application	31
16.16 Suggestions	32
Case 17. Lies, damned lies, and statistics	
Case 18. Database security	
Case 19. Data analysis and cleaning	
Case 20. Database reverse engineering	
Case 21. Database prototyping	
Case 22. Generating a test database	
22.1 Performance evaluation of a database	1
22.2 Building a large test database	2
22.3 Data generation for the ORDERS.db database	3
22.3.1 Data set for CUSTOMER table	3
22.3.2 Data set for the PRODUCT table	10
22.3.3 Data set for the CUSTORDER table	13
22.3.4 Data set for the DETAIL table	15
22.4 Test data generator for ORDERS2.db	18
22.5 What next?	22
Case 23. Database performance	
Case 24. Object/Relational mappers vs Active databases	
Case 25. Text processing	
Case 26. Geographic information systems	
Case 27. Conway's Game of Life	
27.1 Introduction	2
27.2 World and automata representation	3
27.3 Computing the next state	5
27.4 Graphical display of automaton evolution	5
27.5 Generation of SQLdraw scripts.	7
27.6 Packaging the LIFE application	7
27.7 First performance analysis	9
27.8 Optimization 1: indexing tables	10
27.9 Optimization 2: windowing table access	12
27.10 Optimization 3: building a cheap world	14
27.11 Optimization 4: let SQL generate vector graphics	16

- 27.12 Optimization 5: Rewriting the application in Python 17
- 27.13 Some representative Conway's automata 18
- 27.14 Family life 19
- 27.15 Lessons learned 20
- 27.16 The scripts 22
- Case 28. From data bulk loading to database book writing
  - 28.1 Introduction 2
  - 28.2 The database loading problem 2
  - 28.3 Representing the table graph of a database 5
  - 28.4 Topological sorting of the table graph 6
    - 28.4.1 Topological sorting abstract graphs 7
  - 28.5 Coping with non acyclic graphs 8
    - 28.5.1 Non acyclic graphs 8
    - 28.5.2 Checking the acyclicity of a graph 9
    - 28.5.3 The cyclic kernel of a graph 10
  - 28.6 Cyclic vs hard cyclic kernel of a table graph 11
  - 28.7 Contracting the circuits of the hard kernel 13
  - 28.8 Finding the independent circuits of a graph 13
    - 28.8.1 The independent circuits of a graph 13
    - 28.8.2 What is a circuit after all? 14
  - 28.9 Finding the super circuits of a graph 16
    - 28.9.1 Computing the closure of a table graph 16
    - 28.9.2 Finding the reachable set of a node 17
    - 28.9.3 How to denote the super circuits of a graph? 18
    - 28.9.4 Identifying the super circuits of a graph 19
    - 28.9.5 Structure of the solution 21
    - 28.9.6 Revisiting the optional links 22
    - 28.9.7 Intermediate summary 22
  - 28.10 Building the final acyclic graph 23
  - 28.11 Optimizing the final acyclic graph 24
  - 28.12 Loading data: a complex abstract example 26
  - 28.13 The complete data loading generation procedure 30
  - 28.14 Loading the data of a super circuit 31
  - 28.15 Loading data: a real example 33
  - 28.16 Brute force data loading technique 34
  - 28.17 Performance 34
  - 28.18 Complement: extracting the table graph of a database 35
  - 28.19 Topological sorting: application to book development 35

28.20 Other applications 40

28.21 The scripts 41

Case 29. Bill of Material

Case 30. Classifying things (Formal Concept Analysis))

Case 31. Path finders, rovers and Ariadne's thread

31.1 Shortest paths between two cities 1

31.2 Dijkstra's Algorithm 2

31.3 Representation of the graph 6

31.4 Computing the shortest paths from a starting city 8

31.5 Building a general purpose shortest path engine 11

31.6 Ariadne's thread: solving mazes 14

31.7 Exploring a planet 17

31.8 Performance and optimization 20

31.9 The scripts 21

Case 32. Databases and the internet

Case 33. Workflow management

