

A Course in In-Memory Data Management

Hasso Plattner

A Course in In-Memory Data Management

The Inner Mechanics of
In-Memory Databases



Springer

Hasso Plattner
Hasso Plattner Institute
Potsdam, Brandenburg
Germany

ISBN 978-3-642-36523-2 ISBN 978-3-642-36524-9 (eBook)

DOI 10.1007/978-3-642-36524-9

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013932332

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Why We Wrote This Book

Our research group at the HPI has conducted research in the area of in-memory data management for enterprise applications since 2006. The ideas and concepts of a dictionary-encoded column-oriented in-memory database gained much traction due to the success of SAP HANA as the cutting-edge industry product and from followers trying to catch up. As this topic reached a broader audience, we felt the need for proper education in this area. This is of utmost importance as students and developers have to understand the underlying concepts and technology in order to make use of it.

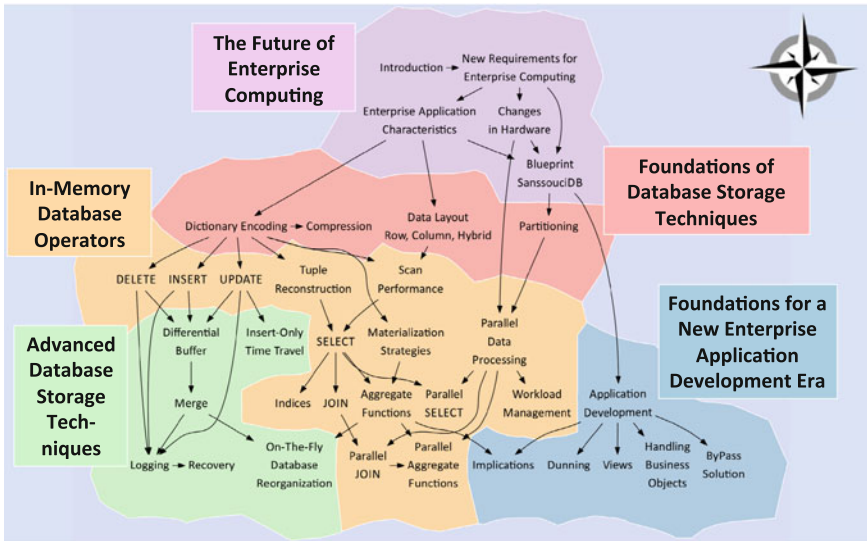
At our institute, we have been teaching in-memory data management in a Master's course since 2009. When I learned about the current movement towards the direction of Massive Open Online Courses, I immediately decided that we should offer our course about in-memory data management to the public. On September 3, 2012 we started our online education with the new online platform <http://www.openHPI.de>. We granted 2,137 graded certificates to the 13,126 participating learners of the first iteration of the online course. Please feel free to register at openHPI.de to be informed about upcoming lectures.

Several thousand people have already used our material in order to study for the homework assignments and final exam of our online course. This book is based on the reading material that we provided to the online community. In addition to that, we incorporated many suggestions for improvement as well as self-test questions and explanations. As a result, we provide you with a textbook teaching you the inner mechanics of a dictionary-encoded column-oriented in-memory database.

Navigating the Chapters

When giving a lecture, content is typically taught in a one-dimensional sequence. You have the advantage that you can read the book according to your interests. To this end, we provide a learning map, which also reappears in the introduction to

make sure that all readers notice it. The learning map shows all chapters of this book, also referred to as learning units, and shows which topics are prerequisites for which other topics. For example, the learning unit “Differential Buffer” ([Chap. 25](#)) is referred to relatively late in the book. Nevertheless, you might already read it earlier. The prerequisites are that you understood the concepts of how “DELETES”, “INSERTs”, and “UPDATES” are conducted without a differential buffer.



The last section of each chapter contains self-test questions. You also find the questions including the solutions and explanations in [Sect. 34.3](#).

The Development Process of the Book

I want to thank the team of our research chair “Enterprise Platform and Integration Concepts” at the Hasso Plattner Institute at the University of Potsdam in Germany. This book would not exist without this team.

Special thanks go to our online lecture core team consisting of *Ralf Teusner*, *Martin Grund*, *Anja Bog*, *Jens Krüger*, and *Jürgen Müller*.

During the preparation of the online lecture as well as during the online lecture itself, the whole research group took care that no email remained unanswered and all reported bugs in the learning material were fixed. Thus, I want to thank the research assistants *Martin Faust*, *Franziska Häger*, *Thomas Kowark*, *Martin Lorenz*, *Stephan Müller*, *Jan Schaffner*, *Matthieu Schapranow*, *David Schwalb*,

Christian Schwarz, Christian Tinnefeld, Arian Treffer, Johannes Wust, as well as our team assistant *Andrea Lange* for their commitment.

During the development process, several HPI bachelor students (Frank Blechschmidt, Maximilian Grundke, Jan Lindemann, Lars Rückert) and HPI master students (Sten Ächtner, Martin Boissier, Ekaterina Gavrilova, Martin Köppelmann, Paul Möller, Michael Wolowyk) supported us during the online lecture preparations. Special thanks go to *Martin Boissier, Maximilian Grundke, Jan Lindemann*, and *Jasper Schulz*, who worked on all the corrections and adjustments that have to be made when teaching material is enhanced in order to print a book.

Help Improving This Book

We are continuously seeking to improve the learning material provided in this book. If you identify any flaws, please do not hesitate to contact me at hasso.plattner@hpi.uni-potsdam.de.

So far, we received bug reports that resulted in improvements in the learning material from the following attentive readers: Shakir Ahmed, Heiko Betzler, Christoph Birkenhauer, Jonas Bränzel, Dmitry Bondarenko, Christian Butzlaff, Peter Dell, Michael Dietz, Michael Max Eibl, Roman Ganopolskyi, Christoph Gilde, Hermann Grahm, Jan Grasshoff, Oliver Hahn, Ralf Hubert, Katja Huschle, Jens C. Ittel, Alfred Jockisch, Ashutosh Jog, Gerold Kasemir, Alexander Kirov, Jennifer Köenig, Stephan Lange, Francois-David Lessard, Verena Lommatsch, Clemens Müller, Hendrik Müller, Debanshu Mukherjee, Holger Pallak, Jelena Perfiljeva, Dieter Rieblinger, Sonja Ritter, Veronika Rodionova, Viacheslav Rodionov, Yannick Rödl, Oliver Roser, Alice-Rosalind Schell, Wolfgang Schill, Leo Schneider, Jürgen Seitz, David Siegel, Markus Steiner, Reinhold Thurner, Florian Tönjes, Wolfgang Weinmann, Bert Wunderlich, and Dieter Zürn.

We are thankful for any kind of feedback and hope that the learning material will be further improved by the in-memory database community.

Hasso Plattner

Contents

1	Introduction	1
1.1	Goals of the Lecture	1
1.2	The Idea	1
1.3	Learning Map	2
1.4	Self Test Questions	3
	References	3

Part I The Future of Enterprise Computing

2	New Requirements for Enterprise Computing	7
2.1	Processing of Event Data	7
2.1.1	Sensor Data	8
2.1.2	Analysis of Game Events	8
2.2	Combination of Structured and Unstructured Data	9
2.2.1	Patient Data	10
2.2.2	Airplane Maintenance Reports	10
2.3	Social Networks and the Web	11
2.4	Operating Cloud Environments	11
2.5	Mobile Applications	12
2.6	Production and Distribution Planning	12
2.6.1	Production Planning	13
2.6.2	Available to Promise Check	13
2.7	Self Test Questions	13
	References	14
3	Enterprise Application Characteristics	15
3.1	Diverse Applications	15
3.2	OLTP Versus OLAP	15
3.3	Drawbacks of the Separation of OLAP from OLTP	16
3.4	The OLTP Versus OLAP Access Pattern Myth	16
3.5	Combining OLTP and OLAP Data	17
3.6	Enterprise Data Characteristics	17

3.7	Self Test Questions	18
	References	18
4	Changes in Hardware	19
4.1	Memory Cells	19
4.2	Memory Hierarchy	20
4.3	Cache Internals	21
4.4	Address Translation	22
4.5	Prefetching	23
4.6	Memory Hierarchy and Latency Numbers	23
4.7	Non-Uniform Memory Architecture	25
4.8	Scaling Main Memory Systems	26
4.9	Remote Direct Memory Access	27
4.10	Self Test Questions	27
	References	28
5	A Blueprint of SanssouciDB	29
5.1	Data Storage in Main Memory	29
5.2	Column-Oriented	29
5.3	Implications of Column-Oriented	30
5.4	Active and Passive Data	31
5.5	Architecture Overview	31
5.6	Self Test Questions	32
	Reference	33

Part II Foundations of Database Storage Techniques

6	Dictionary Encoding	37
6.1	Compression Example	38
6.1.1	Dictionary Encoding Example: First Names	39
6.1.2	Dictionary Encoding Example: Gender	39
6.2	Sorted Dictionaries	40
6.3	Operations on Encoded Values	40
6.4	Self Test Questions	41
7	Compression	43
7.1	Prefix Encoding	43
7.2	Run-Length Encoding	45
7.3	Cluster Encoding	46
7.4	Indirect Encoding	48
7.5	Delta Encoding	51
7.6	Limitations	52
7.7	Self Test Questions	52
	Reference	54

8	Data Layout in Main Memory	55
8.1	Cache Effects on Application Performance	55
8.1.1	The Stride Experiment	55
8.1.2	The Size Experiment.	57
8.2	Row and Columnar Layouts.	58
8.3	Benefits of a Columnar Layout	61
8.4	Hybrid Table Layouts	61
8.5	Self Test Questions.	62
	References	62
9	Partitioning	63
9.1	Definition and Classification	63
9.2	Vertical Partitioning	63
9.3	Horizontal Partitioning	64
9.4	Choosing a Suitable Partitioning Strategy	66
9.5	Self Test Questions.	66
	Reference	67

Part III In-Memory Database Operators

10	Delete	71
10.1	Example of Physical Delete	71
10.2	Self Test Questions.	73
	Reference	73
11	Insert	75
11.1	Example	75
11.1.1	INSERT without New Dictionary Entry	76
11.1.2	INSERT with New Dictionary Entry.	76
11.2	Performance Considerations.	79
11.3	Self Test Questions.	80
12	Update	83
12.1	Update Types.	83
12.1.1	Aggregate Updates	83
12.1.2	Status Updates	84
12.1.3	Value Updates	84
12.2	Update Example.	84
12.3	Self Test Questions.	86
	References	87

13 Tuple Reconstruction	89
13.1 Introduction	89
13.2 Tuple Reconstruction in Row-Oriented Databases.	89
13.3 Tuple Reconstruction in Column-Oriented Databases	90
13.4 Further Examples and Discussion	91
13.5 Self Test Questions.	92
14 Scan Performance	95
14.1 Introduction	95
14.2 Row Layout: Full Table Scan	96
14.3 Row Layout: Stride Access	96
14.4 Columnar Layout: Full Column Scan	97
14.5 Additional Examples and Discussion.	98
14.6 Self Test Questions.	98
15 Select	99
15.1 Relational Algebra	99
15.1.1 Cartesian Product	99
15.1.2 Projection	99
15.1.3 Selection	100
15.2 Data Retrieval	100
15.3 Self Test Questions.	102
16 Materialization Strategies	105
16.1 Aspects of Materialization	105
16.2 Example	106
16.3 Early Materialization.	107
16.4 Late Materialization	108
16.5 Self Test Questions.	111
References	112
17 Parallel Data Processing	113
17.1 Hardware Layer	113
17.1.1 Multi-Core CPUs	114
17.1.2 Single Instruction Multiple Data.	115
17.2 Software Layer.	117
17.2.1 Amdahl's Law	117
17.2.2 Shared Memory	118
17.2.3 Message Passing.	118
17.2.4 MapReduce	119
17.3 Self Test Questions.	120
References	120

18	Indices	121
18.1	Indices: A Query Optimization Approach	121
18.2	Technical Considerations	121
18.3	Inverted Index	123
18.4	Discussion	126
18.4.1	Memory Consumption	126
18.4.2	Lookup Performance	127
18.5	Self Test Questions	129
	Reference	129
19	Join	131
19.1	Join Execution in Main Memory	132
19.2	Hash-Join	133
19.2.1	Example Hash-Join	133
19.3	Sort-Merge Join	135
19.3.1	Example Sort-Merge Join	135
19.4	Choosing a Join Algorithm	136
19.5	Self Test Questions	137
20	Aggregate Functions	141
20.1	Aggregation Example Using the COUNT Function	141
20.2	Self Test Questions	143
21	Parallel Select	145
21.1	Parallelization	145
21.2	Self Test Questions	148
22	Workload Management and Scheduling	149
22.1	The Power of Speed	149
22.2	Scheduling	150
22.3	Mixed Workload Management	150
22.4	Self Test Questions	151
	Reference	152
23	Parallel Join	153
23.1	Partially Parallelized Hash-Join	153
23.2	Parallel Hash-Join	154
23.3	Self Test Questions	155
	References	155
24	Parallel Aggregation	157
24.1	Aggregate Functions Revisited	157
24.2	Parallel Aggregation Using Hashing	158

24.3 Self Test Questions	160
Reference	160

Part IV Advanced Database Storage Techniques

25 Differential Buffer	163
25.1 The Concept	163
25.2 The Implementation	163
25.3 Tuple Lifetime	165
25.4 Self Test Questions	166
References	166
26 Insert-Only	167
26.1 Definition of the Insert-Only Approach	167
26.2 Point Representation	168
26.3 Interval Representation	169
26.4 Concurrency Control: Snapshot Isolation	171
26.5 Insert-Only: Advantages and Challenges	172
26.6 Self Test Questions	173
Reference	174
27 The Merge Process	175
27.1 The Asynchronous Online Merge	176
27.1.1 Prepare Merge Phase	177
27.1.2 Attribute Merge Phase	177
27.1.3 Commit Merge Phase	178
27.2 Exemplary Attribute Merge of a Column	178
27.3 Merge Optimizations	180
27.3.1 Using the Main Store's Dictionary	180
27.3.2 Single Column Merge	181
27.3.3 Unified Table Concept	182
27.4 Self Test Questions	182
References	183
28 Logging	185
28.1 Logging Infrastructure	185
28.2 Logical Versus Dictionary-Encoded Logging	187
28.3 Example	189
28.4 Self Test Questions	191
References	192

29 Recovery	193
29.1 Reading Meta Data	193
29.2 Recovering the Database	194
29.3 Self Test Questions	194
Reference	195
 30 On-the-Fly Database Reorganization	197
30.1 Reorganization in a Row Store	197
30.2 On-the-Fly Reorganization in a Column Store	198
30.3 Excursion: Multi-Tenancy Requires Online Reorganization	199
30.4 Hot and Cold Data	200
30.5 Self Test Questions	201
References	203
 Part V Foundations for a New Enterprise Application Development Era	
 31 Implications on Application Development	207
31.1 Optimizing Application Development for In-Memory Databases	207
31.1.1 Moving Business Logic into the Database	209
31.1.2 Stored Procedures	210
31.1.3 Example Application	211
31.2 Best Practices	212
31.3 Self Test Questions	213
 32 Database Views	215
32.1 Advantages of Views	215
32.2 Layered Views Concept	216
32.3 Development Tools for Views	216
32.4 Self Test Questions	217
References	218
 33 Handling Business Objects	219
33.1 Persisting Business Objects	219
33.2 Object-Relational Mapping	220
33.3 Self Test Questions	221
 34 Bypass Solution	223
34.1 Transition Steps in Detail	224
34.2 Bypass Solution: Conclusion	227
34.3 Self Test Questions	228

Self Test Solutions	229
Glossary	283
Index	295

Abbreviations

ATP	Available-to-Promise
BI	Business Intelligence
ccNUMA	Cache-Coherent Non-Uniform Memory Architecture
CPU	Central Processing Unit
DML	Data Manipulation Language
DPL	Data Prefetch Logic
DRAM	Dynamic Random Access Memory
DW	Data Warehouse
e.g.	For example
EPIC	Enterprise Platform and Integration Concepts
ERP	Enterprise Resource Planning
et al.	And others
etc.	Et cetera
ETL	Extract Transform Load
FSB	Front Side Bus
HDD	Hard Disk Drive
HPI	Hasso-Plattner-Institut
HR	Human resources
i.e.	That is
IMC	Integrated Memory Controller
IMDB	In-Memory Database
IO	Index Offsets
IP	Index Positions
MDX	Multidimensional Expression
MIPS	Million Instructions Per Second
NUMA	Non-Uniform Memory Architecture
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
ORM	Object-Relational Mapping
PADD	Parallel Add
PDA	Personal Digital Assistant
QPI	Quick Path Interconnect
RAM	Random Access Memory

RISC	Reduced Instruction Set Computing
SADD	Scalar Add
SIMD	Single Instruction Multiple Data
SRAM	Static Random Access Memory
SSE	Streaming SIMD Extensions
TLB	Translation Lookaside Buffer
UMA	Uniform Memory Architecture

Figures

Fig. 1.1	Learning map	3
Fig. 2.1	Inversion of corporate structures	12
Fig. 4.1	Memory hierarchy on Intel Nehalem architecture	21
Fig. 4.2	Parts of a memory address	22
Fig. 4.3	Conceptual view of the memory hierarchy	24
Fig. 4.4	(a) Shared FSB, (b) Intel quick path interconnect [Int09].	25
Fig. 4.5	A system consisting of multiple blades	27
Fig. 5.1	Schematic architecture of SanssouciDB	32
Fig. 6.1	Dictionary encoding example	38
Fig. 7.1	Prefix encoding example	44
Fig. 7.2	Run-length encoding example	46
Fig. 7.3	Cluster encoding example	47
Fig. 7.4	Cluster encoding example: no direct access possible	48
Fig. 7.5	Indirect encoding example	49
Fig. 7.6	Indirect encoding example: direct access	50
Fig. 7.7	Delta encoding example	51
Fig. 8.1	Sequential versus random array layout	56
Fig. 8.2	Cycles for cache accesses with increasing stride	57
Fig. 8.3	Cache misses for cache accesses with increasing stride	58
Fig. 8.4	Cycles and cache misses for cache accesses with increasing working sets	59
Fig. 8.5	Illustration of memory accesses for row-based and column-based operations on row and columnar data layouts	60
Fig. 9.1	Vertical partitioning	64
Fig. 9.2	Range partitioning	65
Fig. 9.3	Round robin partitioning	65
Fig. 9.4	Hash-based partitioning	65

Fig. 11.1	Example database table named <i>world_population</i>	76
Fig. 11.2	Initial status of the <i>I name</i> column	77
Fig. 11.3	Position of the string <i>Schulze</i> in the dictionary of the <i>I name</i> column	77
Fig. 11.4	Appending dictionary position of <i>Schulze</i> to the end of the attribute vector	77
Fig. 11.5	Dictionary for first name column	78
Fig. 11.6	Addition of <i>Karen</i> to <i>fname</i> dictionary	78
Fig. 11.7	Resorting the <i>fname</i> dictionary	78
Fig. 11.8	Rebuilding the <i>fname</i> attribute vector	79
Fig. 11.9	Appending the valueID representing <i>Karen</i> to the attribute vector	79
Fig. 12.1	The <i>world_population</i> table before updating	85
Fig. 12.2	Dictionary, old and new attribute vector of the <i>city</i> column, and state of the <i>world_population</i> table after updating	85
Fig. 12.3	Updating the <i>world_population</i> table with a value that is not yet in the dictionary.	86
Fig. 15.1	Example database table <i>world_population</i>	101
Fig. 15.2	Example query execution plan for SELECT statement	101
Fig. 15.3	Execution of the created query plan	102
Fig. 16.1	Example comparison between early and late materialization	106
Fig. 16.2	Example data of table <i>world_population</i>	107
Fig. 16.3	Early materialization: materializing column via dictionary lookups and scanning for predicate	108
Fig. 16.4	Early materialization: scan for constraint and addition to intermediate result.	108
Fig. 16.5	Early materialization: group by <i>ValCity</i> and aggregation	109
Fig. 16.6	Late materialization: lookup predicate values in dictionary	109
Fig. 16.7	Late materialization: scan and logical <i>AND</i>	110
Fig. 16.8	Late materialization: filtering of attribute vector and dictionary lookup	111
Fig. 17.1	Pipelined and data parallelism	114
Fig. 17.2	The ideal hardware?	114
Fig. 17.3	A multi-core processor consisting of 4 cores	115
Fig. 17.4	A server consisting of multiple processors	116
Fig. 17.5	A system consisting of multiple servers.	116
Fig. 17.6	Single instruction multiple data parallelism	117
Fig. 18.1	Example database table named <i>world_population</i>	122
Fig. 18.2	City column of the <i>world_population</i> table	122
Fig. 18.3	Index offset and index positions	123

Fig. 18.4	Query processing using indices: Step 1	124
Fig. 18.5	Query processing using indices: Step 2	124
Fig. 18.6	Query processing using indices: Step 3	125
Fig. 18.7	Query processing using indices: Step 4	125
Fig. 18.8	Query processing using indices: Step 5	125
Fig. 18.9	Attribute vector scan versus index position list read for a column with 30 million entries (note the log-log Scale)	128
Fig. 19.1	The example join tables.	133
Fig. 19.2	Hash map creation	134
Fig. 19.3	Hash-Join phase	135
Fig. 19.4	Building the translation table	136
Fig. 19.5	Matching pairs from both position lists	137
Fig. 20.1	Input relation containing the world population	142
Fig. 20.2	Count example	142
Fig. 21.1	Parallel scan, partitioning each column into 4 chunks	146
Fig. 21.2	Equally partitioned columns	146
Fig. 21.3	Result of parallel scans	147
Fig. 21.4	Result of <i>Positional AND</i>	147
Fig. 21.5	Parallel scans with parallel <i>Positional AND</i>	148
Fig. 23.1	Parallelized hashing phase of a join algorithm	154
Fig. 24.1	Parallel aggregation in SanssouciDB	159
Fig. 25.1	The differential buffer concept	164
Fig. 25.2	Michael Berg moves from Berlin to Potsdam.	165
Fig. 26.1	Snapshot isolation.	172
Fig. 27.1	The concept of the online merge	176
Fig. 27.2	The attribute merge.	178
Fig. 27.3	The attribute in the main store after the merge process	179
Fig. 27.4	The merge process for a column (adapted from [FSKP12])	179
Fig. 27.5	Unified table concept (adapted from [SFL+12]).	182
Fig. 28.1	Logging infrastructure	186
Fig. 28.2	Logical logging	187
Fig. 28.3	Exemplary Zipf distributions for varying alpha values.	188
Fig. 28.4	Cumulated average log size per query for varying value distributions (DC = Dictionary-Compressed)	188
Fig. 28.5	Log size comparison of logical logging and dictionary-encoded logging	189
Fig. 28.6	Example: logging for dictionary-encoded columns	190

Fig. 30.1	Example memory layout for a row store	198
Fig. 30.2	Example memory layout for a column store.	198
Fig. 30.3	Multi-tenancy granularity levels	200
Fig. 30.4	The life cycle of a sales order	201
Fig. 31.1	Three tier enterprise application	208
Fig. 31.2	Comparison of different dunning implementations	212
Fig. 32.1	Using views to simplify join-queries	216
Fig. 32.2	The view layer concept	217
Fig. 33.1	Sales order business object with object data guide representation	220
Fig. 34.1	Initial architecture.	224
Fig. 34.2	Run IMDB in parallel	224
Fig. 34.3	Deploy new applications	225
Fig. 34.4	Traditional data warehouse on IMDB	225
Fig. 34.5	Run OLTP and OLAP on IMDB	226

Tables

Table 4.1 Latency numbers 24

Table 24.1 Possible result for query in listing 24.2 158

Table 26.1 Initial state of example table using point representation 168

Table 26.2 Example table using point representation
after updating the tuple with id = 1 169

Table 26.3 Initial state of example table using interval representation. . . 170

Table 26.4 Example table using interval representation after updating
the tuple with id = 1 170

Table 26.5 Example table using interval representation
to show concurrent updates 171

Table 26.6 Example table using interval representation
to show concurrent updates after first update 172