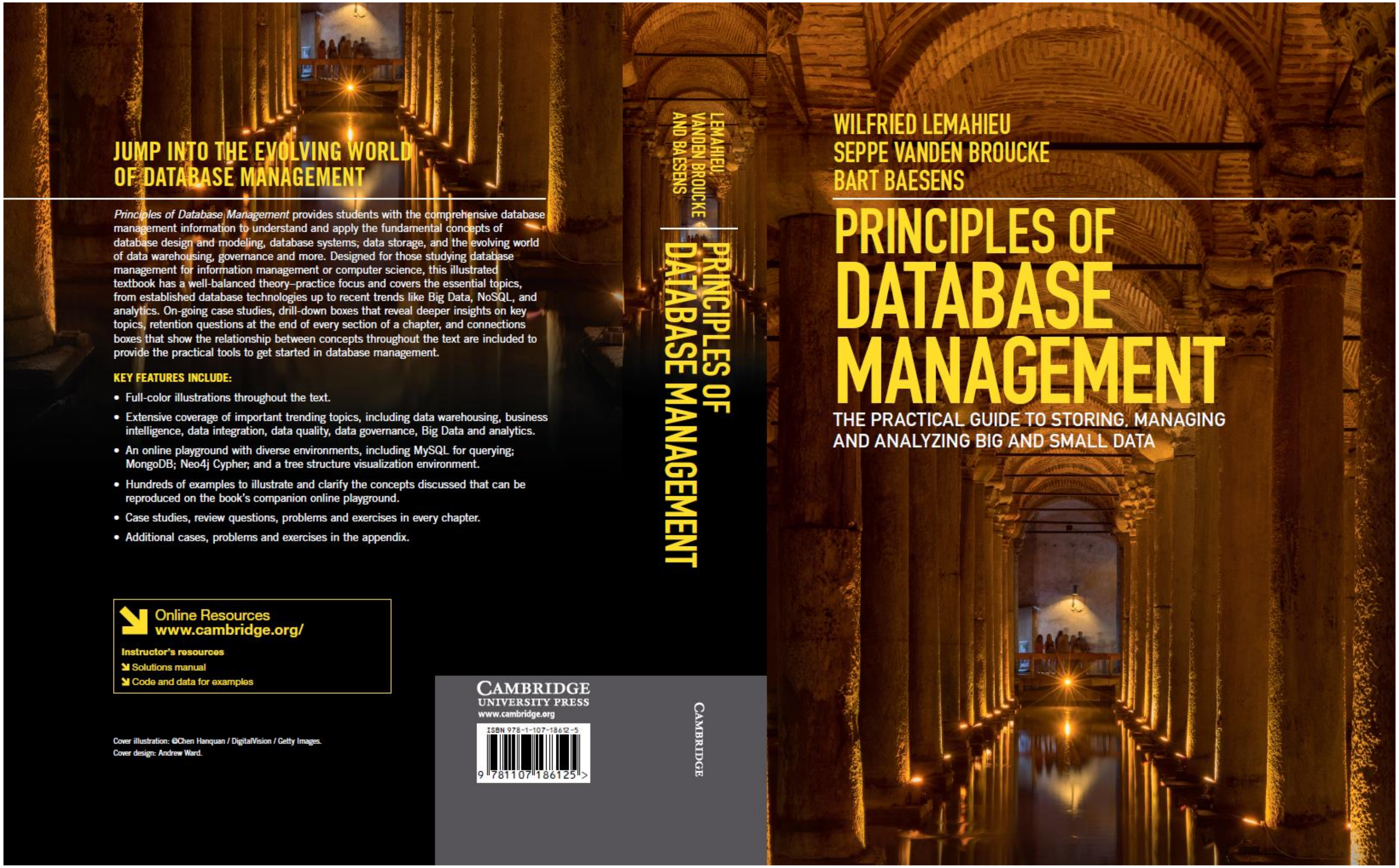
Physical Database Organization



JUMP INTO THE EVOLVING WORLD OF DATABASE MANAGEMENT

Principles of Database Management provides students with the comprehensive database management information to understand and apply the fundamental concepts of database design and modeling, database systems, data storage, and the evolving world of data warehousing, governance and more. Designed for those studying database management for information management or computer science, this illustrated textbook has a well-balanced theory–practice focus and covers the essential topics, from established database technologies up to recent trends like Big Data, NoSQL, and analytics. On-going case studies, drill-down boxes that reveal deeper insights on key topics, retention questions at the end of every section of a chapter, and connections boxes that show the relationship between concepts throughout the text are included to provide the practical tools to get started in database management.

KEY FEATURES INCLUDE:

- Full-color illustrations throughout the text.
- Extensive coverage of important trending topics, including data warehousing, business intelligence, data integration, data quality, data governance, Big Data and analytics.
- An online playground with diverse environments, including MySQL for querying; MongoDB; Neo4j Cypher; and a tree structure visualization environment.
- Hundreds of examples to illustrate and clarify the concepts discussed that can be reproduced on the book's companion online playground.
- Case studies, review questions, problems and exercises in every chapter.
- Additional cases, problems and exercises in the appendix.

Online Resources
www.cambridge.org/

Instructor's resources
Solutions manual
Code and data for examples

Cover illustration: ©Chen Hanquan / DigitalVision / Getty Images.
Cover design: Andrew Ward.

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org
ISBN 978-1-107-18612-5

LEMAHIEU,
VANDEN BROUCKE
AND BAESENS

PRINCIPLES OF
DATABASE MANAGEMENT

CAMBRIDGE

WILFRIED LEMAHIEU
SEPPE VANDEN BROUCKE
BART BAESENS

PRINCIPLES OF
DATABASE
MANAGEMENT

THE PRACTICAL GUIDE TO STORING, MANAGING
AND ANALYZING BIG AND SMALL DATA

www.pdbmbook.com

Introduction

- Physical Database Organization and Database Access Methods
- Enterprise Storage Subsystems and Business Continuity

Physical Database Organization and Database Access Methods

- From Database to Tablespace
- Index Design
- Database Access Methods
- Join Implementations

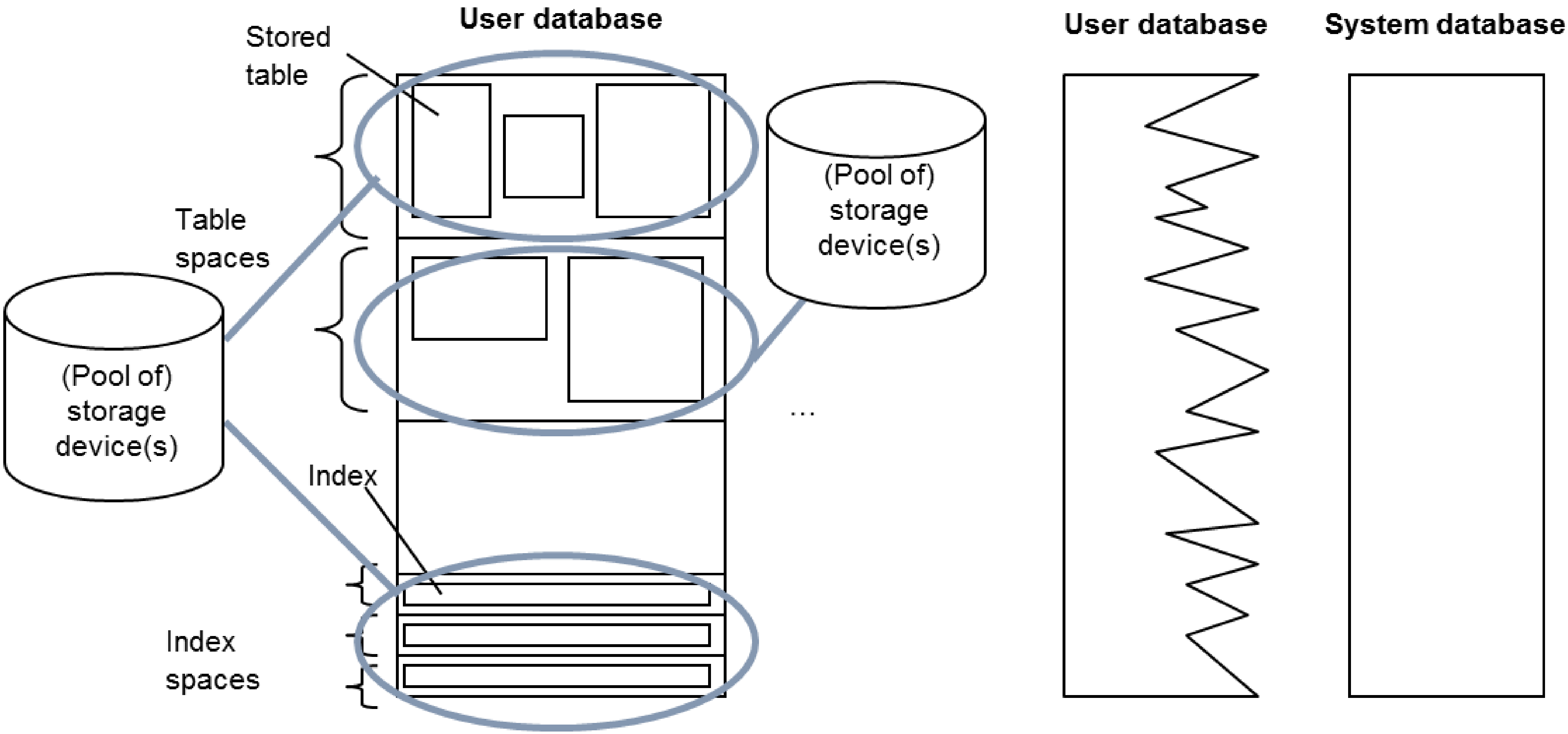
From Database to Tablespace

- Physical user database: collection of index files and data files
- Tablespace: physical container of database objects
  - 1 or more physical files possibly distributed over multiple storage devices
  - every logical table is assigned to a tablespace to be persisted physically (stored table)
  - a stored table occupies one or more disk blocks or pages in the tablespace
  - can contain indexes as well

From Database to Tablespace

- Use of tablespaces has an impact on physical database design
  - intra query parallelism: different subsets of data can be searched in parallel for a single, complex query
  - inter query parallelism: many simple queries executed in parallel
- Physical database design comes down to attributing logical concepts to physical constructs
  - joint responsibility of database designer and DBA

From Database to Tablespace



Index Design

- Indexes important tuning instrument to database designer and DBA

Index type	**Impacts physical ordering of tuples**	**Unique search key**	**Dense or sparse**
Primary	Yes	Yes	Sparse
Clustered	Yes	No	Dense or sparse
Secondary	No	Yes	Dense
		No	Dense or inverted file

Index Design

- Reasons for index creation
  - efficient retrieval of rows according to certain queries or selection criteria
  - efficient performance of join queries
  - enforce uniqueness on a column value or combination of column values
  - logical or physical ordering of rows in table

Index Design

- No standard SQL syntax for index creation
- Common syntax

```
CREATE [UNIQUE] INDEX INDEX_NAME
ON TABLE_NAME (COLUMN_NAME [ORDER]
{, COLUMN_NAME [ORDER]})
[CLUSTER]
```

Index Design

```
CREATE UNIQUE INDEX PRODNR_INDEX
ON PRODUCT(PRODNR ASC)

CREATE INDEX PRODUCTDATA_INDEX
ON PRODUCT(PRODPRICE DESC, PRODTYPE ASC)

CREATE INDEX PRODNAME_INDEX
ON PRODUCT(PRODNAME ASC)
CLUSTER

CREATE INDEX PRODSUPPLIER_INDEX
ON PRODUCT(SUPPLIERNR ASC)
```

Index Design

- Only 1 primary or clustered index per table but as many secondary indexes as desired
- Cost of index:
  - storage capacity
  - update overhead
- Physical data independence: indexes can be added or deleted without affecting logical data model or applications
- Secondary indexes can be constructed or removed without affecting actual data files

Database Access Methods

- Functioning of Query Optimizer
- Index Search (with Atomic Search Key)
- Multiple Index and Multicolumn Index Search
- Index Only Access
- Full Table Scan

Functioning of Query Optimizer

- SQL is declarative query language
- Different access paths exist to same data, but their time varies
- Cost-based optimizers calculate optimal access plan according to set of built-in cost formulas as well table(s) involved in query, available indexes, statistical properties of data in tables, etc.
- Query processor assists in execution of queries and consists of DML compiler, query parser, query rewriter, query optimizer and query executor

Functioning of Query Optimizer

- DBMS maintains following data in catalog
  - Table related data
    - number of rows, number of disk blocks occupied by table, number of overflow records associated with table
  - Column related data
    - number of different column values, distribution of column values
  - Index related data
    - number of different values for indexed search keys and for individual attribute types of composite search keys, number of disk blocks occupied by index, index type (primary/clustered or secondary)
  - Tablespace related data
    - number and size of tables in tablespace, device specific I/O properties of device on which table resides

Functioning of Query Optimizer

- Filter Factor (FF): fraction of total number of rows expected to satisfy query predicate associated with attribute type (e.g., CustomerID = 11349, Gender = M, Year of Birth ≥ 1970)
- For queries over a single table, expected query cardinality (QC) equals table cardinality (TC) multiplied by product of filter factors of respective search predicates in query:
  $QC = TC \times FF_1 \times FF_2 \times \ldots FF_n$
- Default estimate for FF_i is $1/NV_i$, with NV_i representing number of different values of attribute type A_i

Functioning of Query Optimizer

- Example: Customer Table
- 10000 rows (TC = 10000)
- Query:

```
SELECT CUSTOMERID
FROM CUSTOMERTABLE
WHERE COUNTRY = 'U.K.'
AND GENDER = 'M'
```

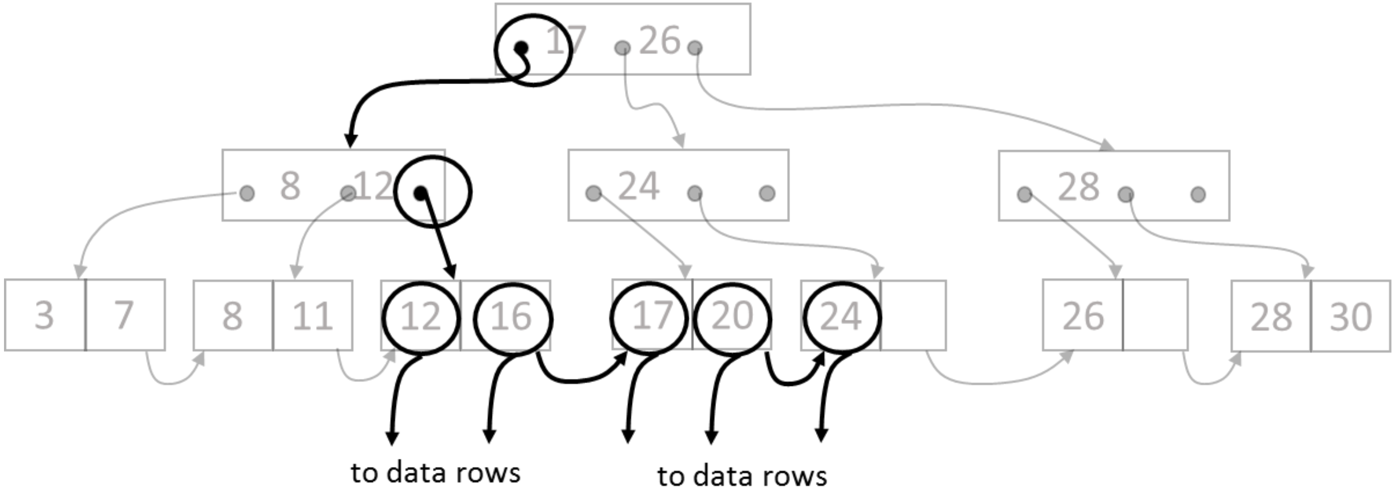
- Assume 20 countries and 2 genders: $FF_{Country} = 0,05$ and $FF_{Gender} = 0,5$
- QC = 10000 x 0,05 x 0,5 = 250

Index Search (with Atomic Search Key)

- Single query predicate where query involves search key with only single attribute type

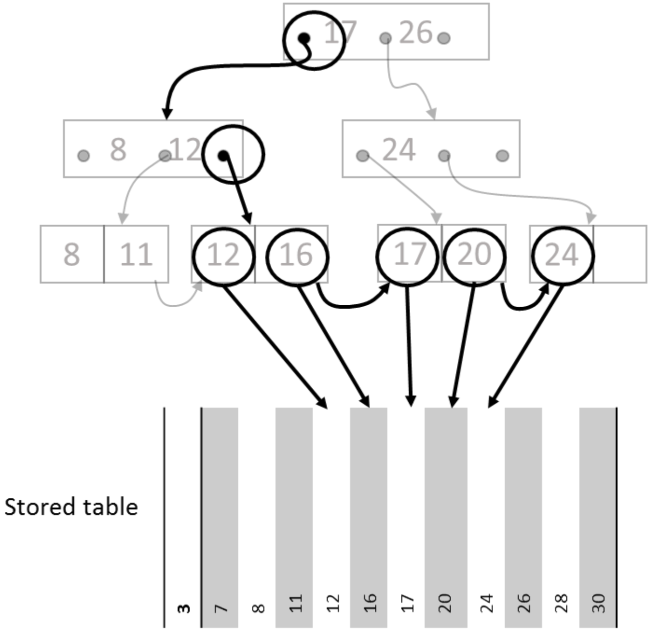
```
SELECT *
FROM MY_TABLE
WHERE MY_KEY >= 12
AND MY_KEY <= 24
```

- Index search using B⁺-tree

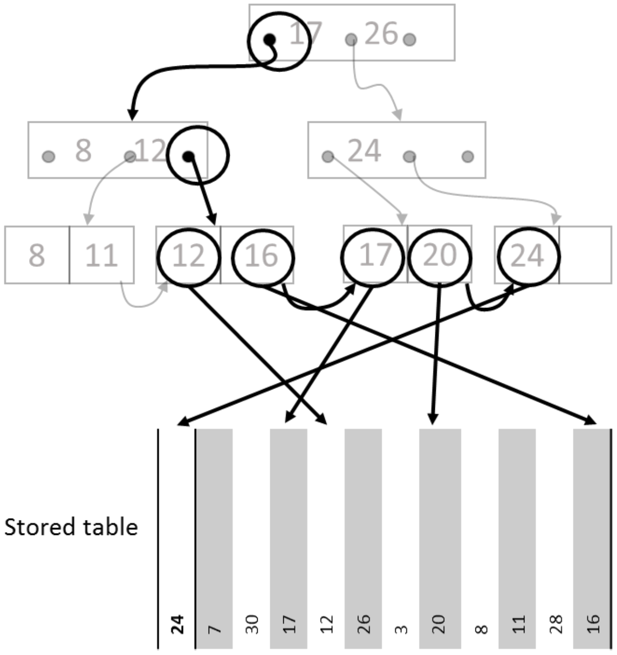


Index Search (with Atomic Search Key)

- If query involves only single search key value, then hashing would be efficient alternative
- As to range queries, primary or clustered index is even more efficient than secondary index (sba versus rba)



Primary index



Secondary index

Multiple Index and Multicolumn Index Search

- If search key is composite and indexed attribute types are same as attribute types in search key, then multicolumn index allows filtering out and retrieving only data rows that satisfy query
- Search key with 5 attribute types, each with 10 possible values, gives multicolumn index with 100,000 (=10^5) entries
  - when using single column indexes instead, there would be 5 indexes, each with 10 entries (total: 50)
- For queries involving arbitrary subset of attribute types, multicolumn index should have all attribute types involved in query predicates in its leftmost columns

Multiple Index and Multicolumn Index Search

- To cater for all possible search keys with n attribute types or less, $\binom{n}{\lceil n/2 \rceil}$ indexes are required
- With 3 attribute types, $\binom{3}{2}$ = 3 indexes are required. Examples:
  - – A query involving A_1, A_2 and A_3 → any index
  - – A query involving A_1 and A_3 → use (A_3, A_1, A_2)
  - – A query involving A_2 and A_3 → use (A_2, A_3, A_1)
  - – A query involving only A_1 → use (A_1, A_2, A_3)

Multiple Index and Multicolumn Index Search

- Multicolumn indexes only appropriate for selective cases
  - E.g., queries executed very often or very time critical
- Alternative is multiple single column indexes, or indexes with fewer columns
- Example:

SELECT *

FROM MY_TABLE

WHERE A_1 = $VALUE_1$

AND A_2 = $VALUE_2$

AND …

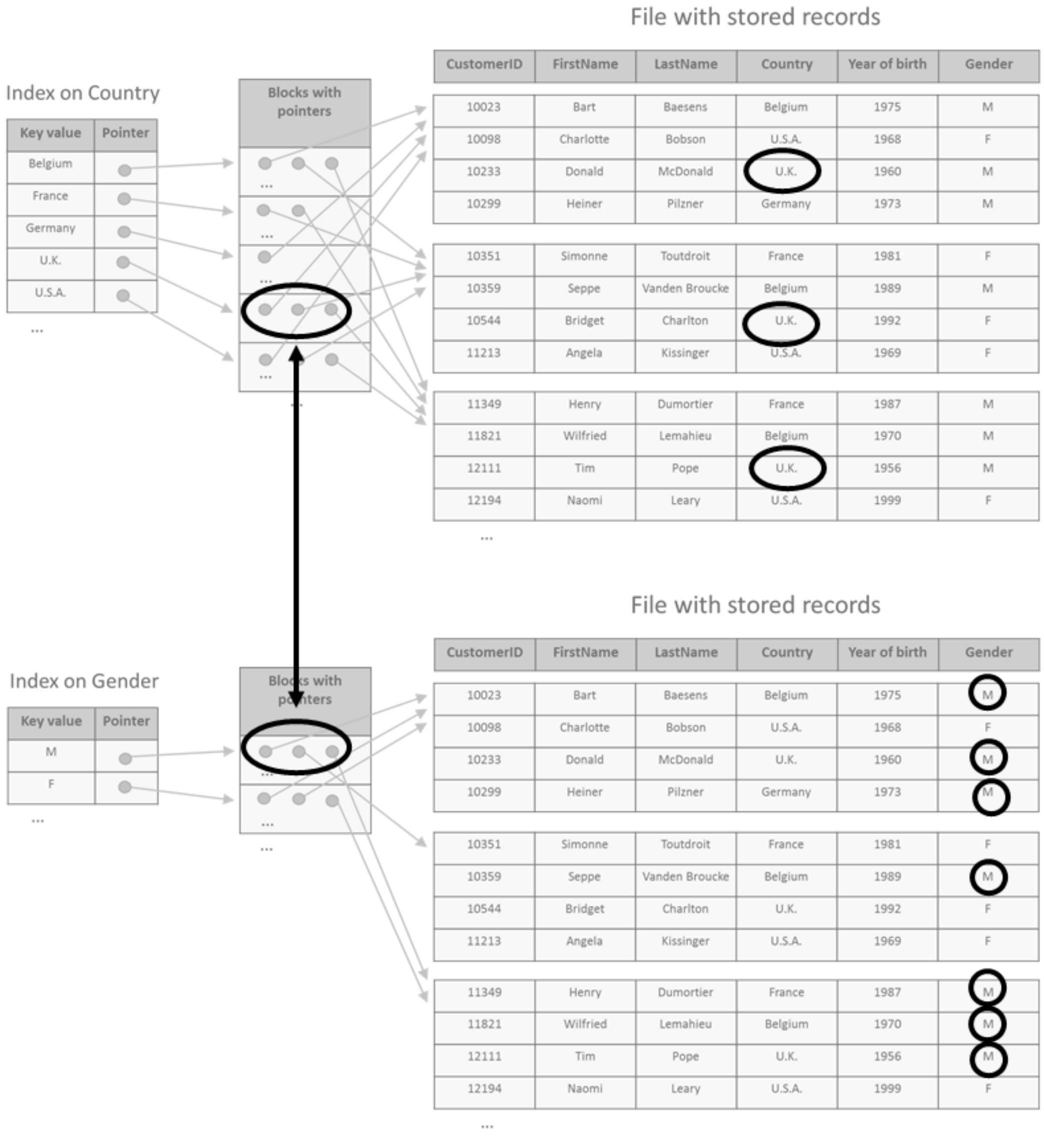
AND A_n = $VALUE_n$

Take intersection between sets of pointers in index entries that correspond to desired values of A_i

Note: Take union in case of OR!

Multiple Index and Multicolumn Index Search

```
SELECT CUSTOMERID
FROM CUSTOMERTABLE
WHERE COUNTRY = 'U.K.'
AND GENDER = 'M'
```



Key value	Pointer
Belgium	●
France	●
Germany	●
U.K.	●
U.S.A.	●
…	

File with stored records

CustomerID	FirstName	LastName	Country	Year of birth	Gender
10023	Bart	Baesens	Belgium	1975	M
10098	Charlotte	Bobson	U.S.A.	1968	F
10233	Donald	McDonald	U.K.	1960	M
10299	Heiner	Pilzner	Germany	1973	M
10351	Simonne	Toutdroit	France	1981	F
10359	Seppe	Vanden Broucke	Belgium	1989	M
10544	Bridget	Charlton	U.K.	1992	F
11213	Angela	Kissinger	U.S.A.	1969	F
11349	Henry	Dumortier	France	1987	M
11821	Wilfried	Lemahieu	Belgium	1970	M
12111	Tim	Pope	U.K.	1956	M
12194	Naomi	Leary	U.S.A.	1999	F
…					

Key value	Pointer
M	●
F	●
…	

File with stored records

CustomerID	FirstName	LastName	Country	Year of birth	Gender
10023	Bart	Baesens	Belgium	1975	M
10098	Charlotte	Bobson	U.S.A.	1968	F
10233	Donald	McDonald	U.K.	1960	M
10299	Heiner	Pilzner	Germany	1973	M
10351	Simonne	Toutdroit	France	1981	F
10359	Seppe	Vanden Broucke	Belgium	1989	M
10544	Bridget	Charlton	U.K.	1992	F
11213	Angela	Kissinger	U.S.A.	1969	F
11349	Henry	Dumortier	France	1987	M
11821	Wilfried	Lemahieu	Belgium	1970	M
12111	Tim	Pope	U.K.	1956	M
12194	Naomi	Leary	U.S.A.	1999	F
…					

Multiple Index and Multicolumn Index Search

- For queries with many predicates, multicolumn index over all predicates is most efficient: FF is equal to query's FF
  - unfeasible if many attribute types are used in query predicates
- As an alternative multiple indexes can be combined
- The more selective a query predicate's FF, the more desirable it is to use index on corresponding attribute type
- Number of block accesses determines performance, not number of rows retrieved
  - using primary index or clustered index may be more efficient than using secondary index, especially for range queries

Index Only Access

- Optimizer might be 'lucky' such that query can be executed based solely on information in index
- Example

```
SELECT LASTNAME
FROM CUSTOMERTABLE
WHERE COUNTRY = 'U.K.'
AND GENDER = 'M'
```

- Index only access if there exists a multicolumn index, or a combination of single column indexes, over attribute types LastName, Country and Gender
- Note: the more attribute types are included in the index, the higher the negative performance impact of update queries!

Full Table Scan

- If no index is available then need to linearly search entire table
- For very small tables, or for queries that require nearly all of a table’s tuples anyway, full table scan might be more efficient
- The higher query’s FF and/or the larger the table, the less efficient a full table scan will be

Join Implementations

- Recap
- Nested-Loop join
- Sort-Merge join
- Hash join

Recap

- Database organization should also consider join queries
- General notation of an inner join between tables R and S :

$$R \underset{r(a)\,\theta\,s(b)}{\bowtie} S$$

- θ-operator specifies join condition

Recap

Table R

Employee	Payscale
Cooper	1
Gallup	2
O'Donnell	1
Smith	2

Table S

Payscale	Salary
1	10000
2	20000

$R \bowtie S$

r(payscale) = s(payscale)

Employee	Payscale	Salary
Cooper	1	10000
Gallup	2	20000
O'Donnell	1	10000
Smith	2	20000

Nested-Loop Join

- One table is denoted as inner table and other outer table
- For every row in outer table, all rows of inner table are retrieved and compared to current row of outer table
  - if join condition satisfied, both rows are joined and put in output buffer
- Inner table traversed as many times as there are rows in outer table
- Mainly effective if
  - inner table is very small or if internal data model provides facilities for efficient access to inner table
  - FF of other predicates is very restrictive with respect to rows that qualify in inner table

Nested-Loop Join

$$R \bowtie_{r(a) = s(b)} S$$

Denote S → outer table

```
For every row s in S do
     {for every row r in R do
          {if r(a) = s(b) then join r with s and place in output buffer}
     }
```

Sort-Merge Join

- Tuples in both tables first sorted according to attribute types in join condition
- Both tables traversed in this order, with rows that satisfy join condition combined and put in output buffer
- Every table traversed only once (↔Nested-Loop Join)
- Appropriate if many rows in both tables satisfy query predicates and/or if no indexes over join columns

Sort-Merge Join

$$R \underset{r(a) = s(b)}{\bowtie} S$$

```
Stage 1:  sort R according to r(a)
          sort S according to s(b)

Stage 2: retrieve the first row r of R
         retrieve the first row s of S
         for every row r in R
            {while s(b) < r(a)
              read the next row s of S
              if r(a) = s(b) then join r with s and place in output buffer}
```

Hash Join

- Hashing algorithm applied to join attribute type(s) for table R
  - corresponding rows assigned to buckets in a hash file
- Same hashing algorithm applied to join attribute type(s) of second table S
- If hash value for S refers to non-empty bucket in hash file, corresponding rows of R and S are compared according to join condition.
- If join condition satisfied, rows of R and S are joined and put in output buffer
- Performance depends on size of hash file and whether it can be kept in internal memory

Enterprise Storage Subsystems and Business Continuity

- Disk Arrays and RAID
- Enterprise Storage Subsystems
- Business Continuity

Disk Arrays and RAID

- Storage capacity of hard disk drives increased but performance stalled
- More efficient to combine multiple smaller physical disk drives into one larger logical drive
  - distributing data over multiple physical drives allows for parallel retrieval
  - data redundancy introduced, mitigating risk of failure

Disk Arrays and RAID

- RAID is a technology in which standard HDDs are coupled to dedicated hard disk controller (the RAID controller) to make them appear as single logical drive

Disk Arrays and RAID

- Techniques applied in RAID
  - Data striping
    - subsections of data file (strips) distributed over multiple disks to be read and written in parallel
    - with n disks, bit or block i is written to disk (i mod n) + 1
    - with bit level data striping, a byte is split into 8 individual bits, to be distributed over available disks
    - with block level data striping, each block is stored in its entirety on single disk, but respective blocks of same file are distributed over disks

Disk Arrays and RAID

- Techniques applied in RAID (contd.)
  - Redundancy
    - redundant data stored to increase reliability
    - E.g., error detection and correcting codes such as Hamming codes or parity bits
  - Disk mirroring
    - for each disk there is an exact copy (mirror) containing same data
    - mirroring consumes much more storage space than error correcting codes

Disk Arrays and RAID

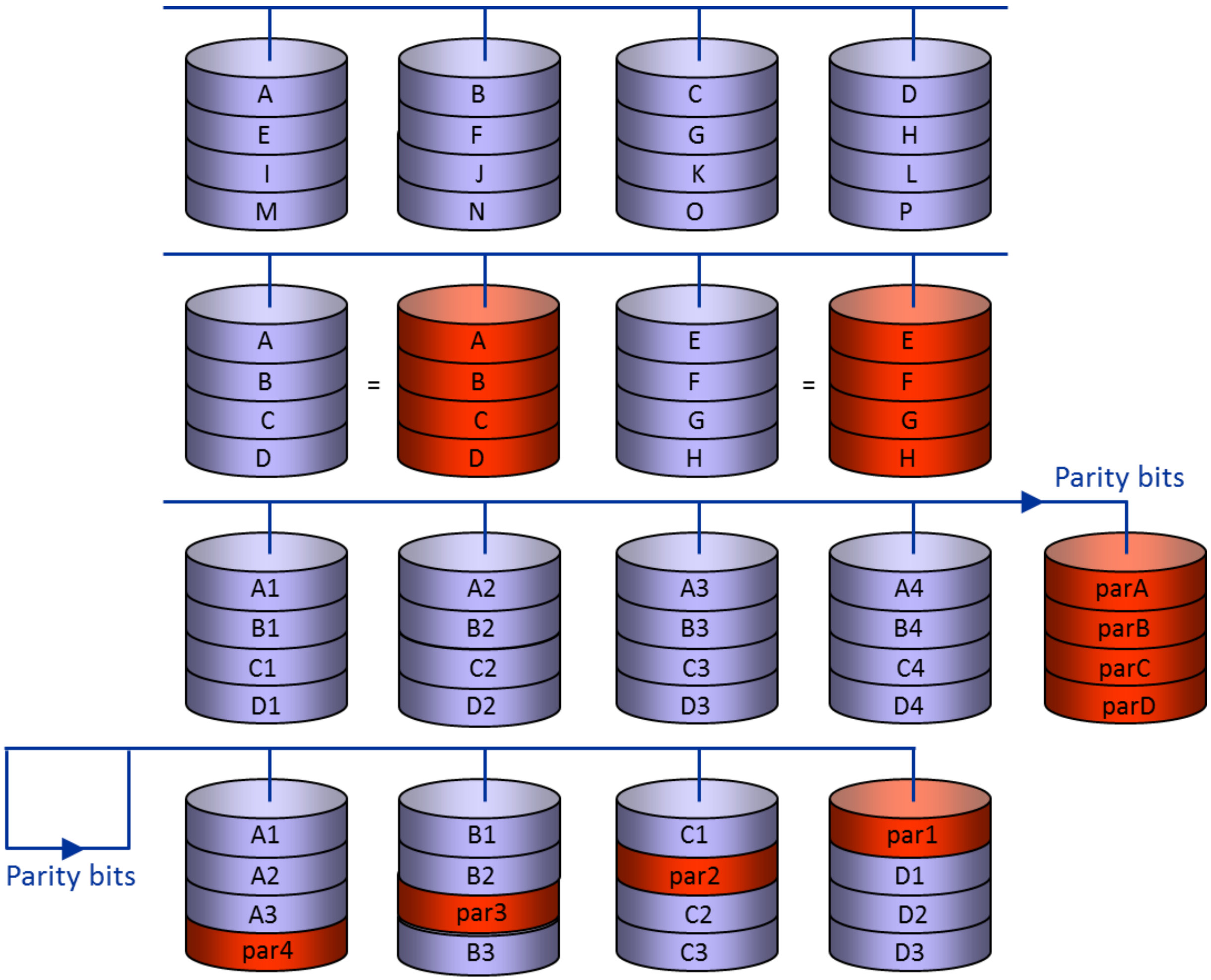
- Multiple RAID configurations (RAID levels)

RAID Level	Description	Fault tolerance	Performance
0	Block level striping	No error correction	Improved read and write performance due to parallelism (multiple processes can read individual blocks in parallel)
1	Disk mirroring	Error correction due to complete duplication of data	Improved read performance: both disks can be accessed by different processes in parallel Write performance is slightly worse, since data needs to be written twice
2	Bit level striping, with separate checksum disk	Error correction through Hamming codes	Improved read performance through parallelism Slower write performance: calculation of checksum; checksum disk involved in every write may become a bottleneck

Disk Arrays and RAID

RAID Level	Description	Fault tolerance	Performance
3	Bit level striping, with parity bits on separate parity disk	Error correction through parity bits	Improved read performance through parallelism, esp. for large, sequential transfers Slower write performance: less calculation overhead than with RAID 2, but parity disk involved in every write may still become a bottleneck
4	Block level striping, with parity bits on separate parity disk	Error correction through parity bits	No improved read performance for individual blocks, but support for efficient parallel block accesses Slower write performance; see RAID 3
5	Block level striping, with distributed parity bits	Error correction through parity bits	Read performance: see RAID 4 Better write performance than RAID 4: parity bits distributed over data disks, so no parity drive as bottleneck

Disk Arrays and RAID



RAID 0
Block level striping

RAID 1
Mirroring

RAID 3
Block level striping
(separate parity disk)

RAID 5
Block level striping
(parity bits between data)

Disk Arrays and RAID

- RAID level 0 used if performance more important than fault tolerance
- RAID level 1 mostly used for very critical data (e.g. logfile of DBMS)
- RAID level 5 quite popular overall, as it strikes a balance between read and write performance, storage efficiency and fault tolerance

Enterprise Storage Subsystems

- Overview and Classification
- DAS (Directly Attached Storage)
- SAN (Storage Area Network)
- NAS (Network Attached Storage)
- NAS Gateway
- iSCSI / Storage over IP

Overview and Classification

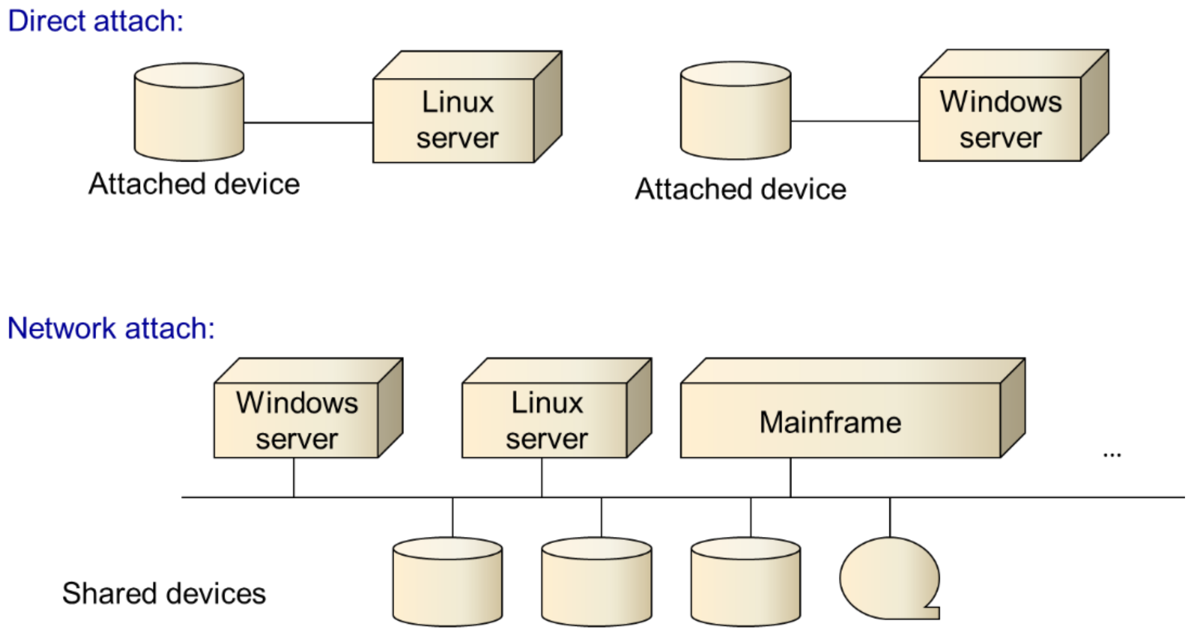
- Starting insights
  - management cost of storage exceeds purchasing cost
  - need for more flexible, distributed storage architectures, which still offer centralized management capabilities
- Modern enterprise storage subsystems often involve networked storage
- Network topology provides for transparent any-to-any connectivity

Overview and Classification

- Characteristics
  - accommodate for high performance data storage
  - cater for scalability
  - support data distribution and location transparency
  - support interoperability and data sharing
  - reliability and near continuous availability
  - protection against hard- and software malfunctions and data loss as well as hackers
  - improve manageability and reduce management cost

Overview and Classification

- Classification according to connectivity, medium and I/O protocol
- Connectivity
  - refers to how storage devices are connected to processors and/or servers
  - direct versus network



Overview and Classification

- Medium refers to physical cabling and low-level protocol
  - – SCSI (Small Computer Systems Interface)
    - used for high-performance and high-capacity workstations and servers
    - 2 elements: command set to communicate with storage devices and specifications for low-level protocol and cabling
  - – Ethernet
    - long-standing standard medium for LANs and WANs
  - – Fibre Channel (FC)
    - connect high-end storage systems to servers
    - fiber optical cable or copper wire

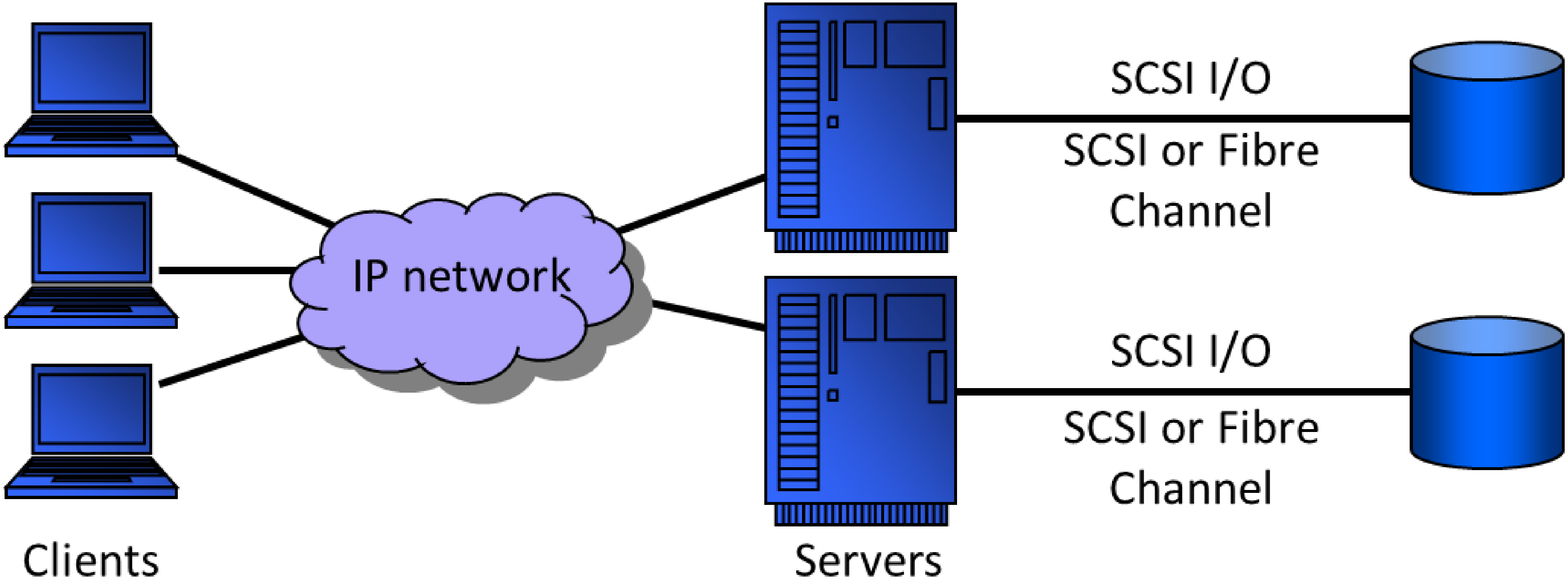
Overview and Classification

- I/O protocol denotes command set to communicate with storage device
  - Block level I/O protocols
    - I/O commands defined at level of requests for individual blocks
    - can be based on SCSI command set
  - File level I/O protocols
    - commands defined at level of requests for entire files
    - protocol is device independent
    - E.g., Network File System (NFS, UNIX), Common Internet File System (CIFS, Windows), HTTP, FTP

DAS (Directly Attached Storage)

- Storage devices directly connected to individual servers
- Block level I/O protocol used
- Medium can be standard SCSI cable, fibre channel, Ethernet cable
- No network for traffic between servers and storage devices
- Simplest and least expensive solution
- No capabilities for centralized storage management and sharing unused disk capacity
- Vulnerable to hardware failures

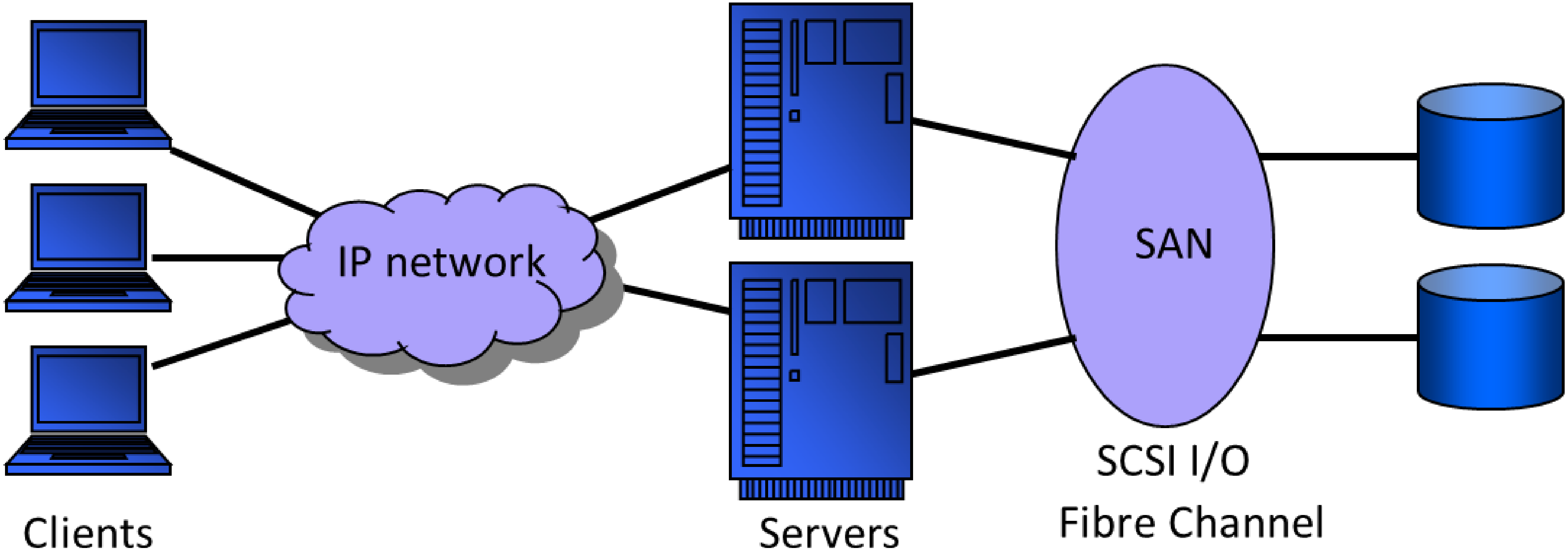
DAS (Directly Attached Storage)



SAN (Storage Area Network)

- Storage related data transfer occurs over a dedicated network
- Servers and storage devices communicate using block level I/O protocol
- Network provides any-to-any connectivity between servers and storage device
- Fibre Channel commonly used as medium
- Clients and servers communicate over IP-based LAN or WAN
- Superior to DAS in terms of availability (data sharing)
- Best solution in terms of performance
- Also offers flexibility and scalability
- However, SAN complex and expensive with ever evolving standards

SAN (Storage Area Network)



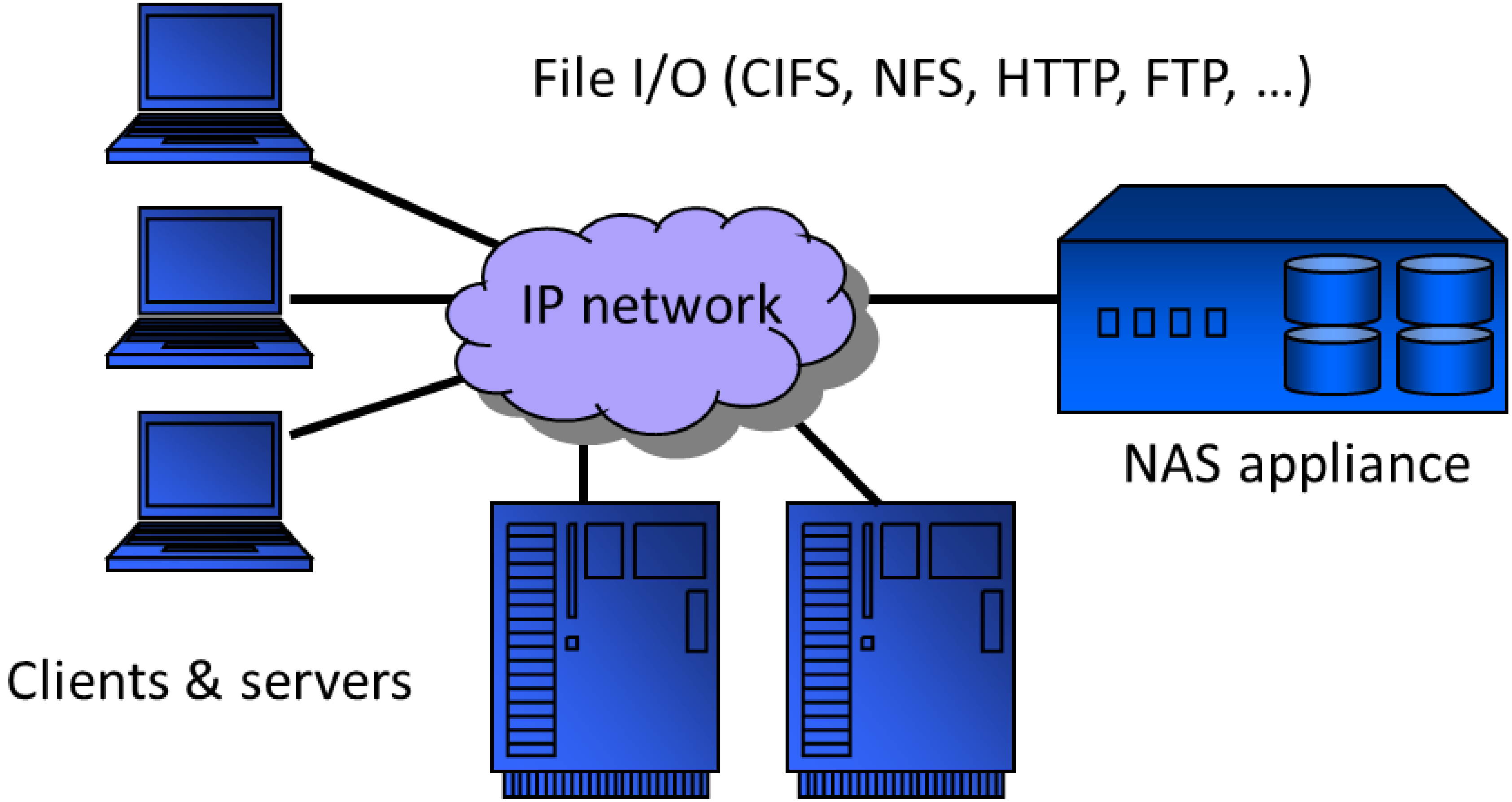
NAS (Network Attached Storage)

- NAS device also called ‘NAS appliance’
  - specialized device which can be ‘plugged’ straightforwardly into TCP/IP based LAN or WAN (Ethernet)
  - stripped-down file server with combination of processor, operating system and set of hard disk drives (no keyboard or screen)
  - less expensive and complex
  - accessed through file level I/O protocol

NAS (Network Attached Storage)

- NAS offers file system to network, with file requests translated by NAS’s internal processor into SCSI block I/O commands
- NAS offers flexible, simple and inexpensive facilities to add additional storage in ‘plug and play’ fashion
- Performance typically lower than with SAN
- Indirection of file level access translated into block level access is less efficient

NAS (Network Attached Storage)



File I/O (CIFS, NFS, HTTP, FTP, ...)

IP network

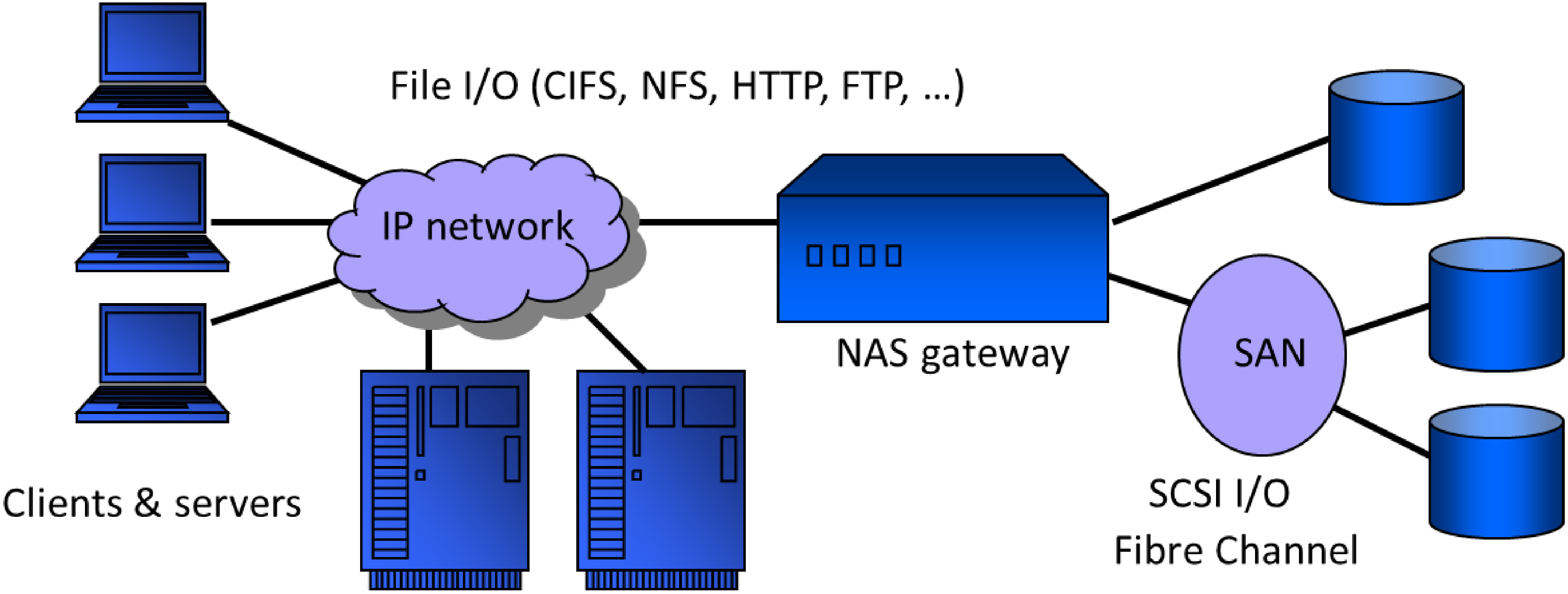
NAS appliance

Clients & servers

NAS Gateway

- Similar to NAS device, but without hard disk drives; only processor and stripped-down operating system
- Can be plugged into a TCP/IP based LAN or WAN on one side and connected to external disk drives on other side (using e.g., DAS or SAN)
- More flexibility and scalability than normal NAS device
- Allows plugging existing disk array into LAN or WAN
- Can yield hybrid NAS/SAN environment

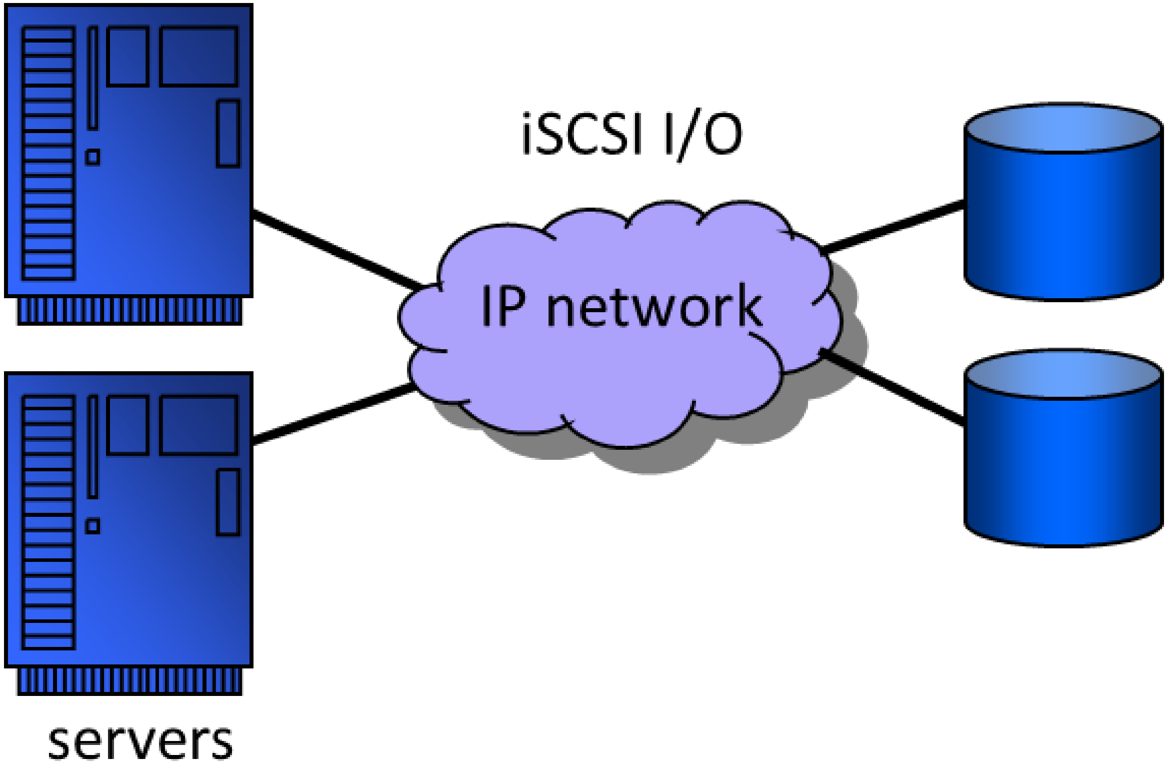
NAS Gateway


File I/O (CIFS, NFS, HTTP, FTP, …)
IP network
NAS gateway
SAN
Clients & servers
SCSI I/O
Fibre Channel

iSCSI / Storage over IP

- iSCSI (a.k.a. Internet SCSI, storage over IP) is similar to SAN but based on Ethernet (instead of Fibre Channel)
- SCSI block level I/O commands packaged and sent over TCP/IP network (LAN or WAN)
- In between SAN and NAS: block level disk access like SAN, but Ethernet based like NAS
- Provides lower cost alternative to SANs
- Popular in small and medium organizations
- Can cover larger distances than FC-based SANs, but typically slower than Fibre Channel

iSCSI / Storage over IP



iSCSI / Storage over IP

Technology	Connectivity	Medium	I/O Protocol
DAS	Direct attach	SCSI cable, Point-to-point FC (or Ethernet)	SCSI block level I/O
SAN	Network attach	FC	SCSI block level I/O
NAS + NAS	Network	Ethernet	File level I/O

Business Continuity

- Introduction
- Contingency Planning, Recovery Point and Recovery Time
- Availability and Accessibility of Storage Devices
- Availability of Database Functionality
- Data Availability

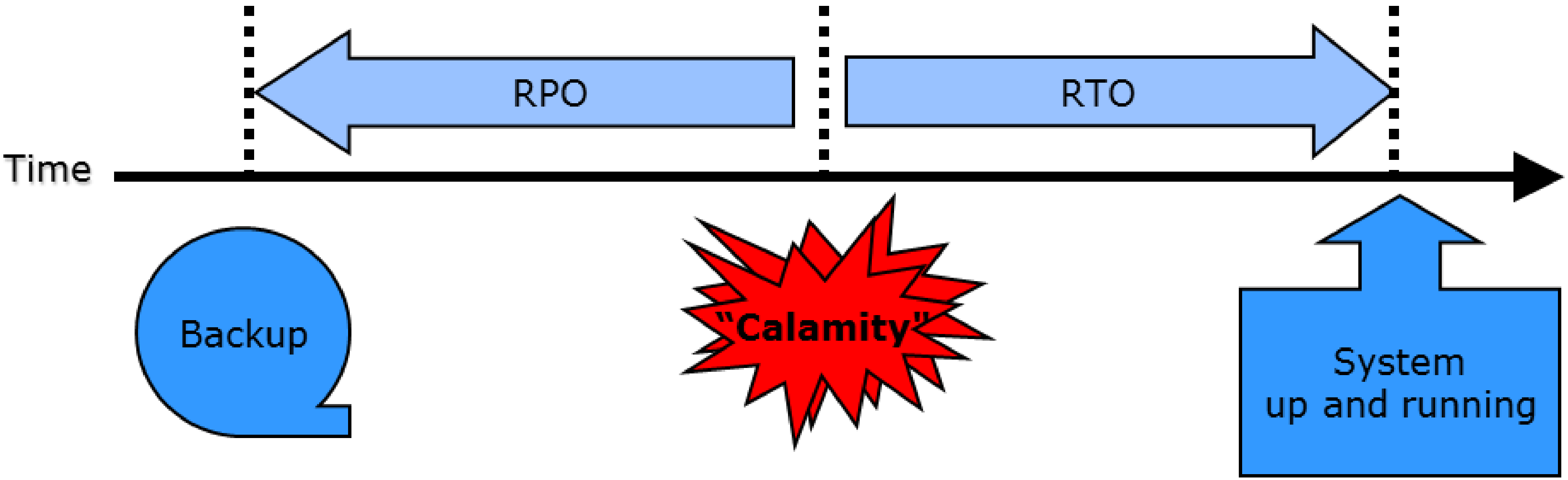
Introduction

- Business continuity: an organization’s ability to guarantee its uninterrupted functioning, despite possible planned or unplanned downtime of the hard- and software supporting its database functionality
- Planned downtime due to backups, maintenance, upgrades, etc.
- Unplanned downtime due to malfunctioning of hardware or software
- Disaster tolerance considers an organization’s endurance against human or nature induced disasters

Contingency Planning, Recovery Point and Recovery Time

- Contingency plan formulates an organization’s measures with respect to business continuity and recovery
- Quantification of recovery objectives
  - Recovery Time Objective (RTO) specifies amount of downtime that is acceptable, after calamity
  - Recovery Point Objective (RPO) specifies degree to which data loss is acceptable after calamity

Contingency Planning, Recovery Point and Recovery Time



Contingency Planning, Recovery Point and Recovery Time

- Aim of contingency plan is to minimize RPO and/or RTO
- Avoid single points of failure
  - availability and accessibility of storage devices
  - availability of database functionality
  - availability of data itself

Availability and Accessibility of Storage Devices

- Networked storage avoids single points of failure that a DAS setup implies with respect to connectivity between servers and storage devices
- Different RAID levels not only impact RPO, but also RTO
  - mirror set-up in RAID 1 allows for uninterrupted storage device access
  - redundancy by parity bits in other RAID levels requires some time to reconstruct the data

Availability of Database Functionality

- First approach is manual failover by means of spare server with DBMS software
  - negative impact on RTO
- More complex approach is clustering which refers to multiple interconnected computer systems (nodes) working together as unity
  - improve performance by means of parallelism and/or availability through redundancy
  - automated failover where nodes in cluster take over workload of failing node
  - rolling upgrades
  - better impact on RTO

Data Availability

- Safeguard data by means of backup and/or replication
- Approaches
  - tape backup
    - database copied periodically to tape storage
    - least expensive but time consuming
    - also restoring is time consuming
    - negative impact on RTO and RPO

Data Availability

- Approaches (contd.)
  - hard disk backup
    - more efficient than tape backup
    - positive impact on RTO and RPO
  - electronic vaulting
    - safeguard backup copies at sufficient distance from primary site to avoid them both being involved in same incident
    - backup data is transmitted over a network to hard disk or tape devices at secure vaulting facility or at alternate data center

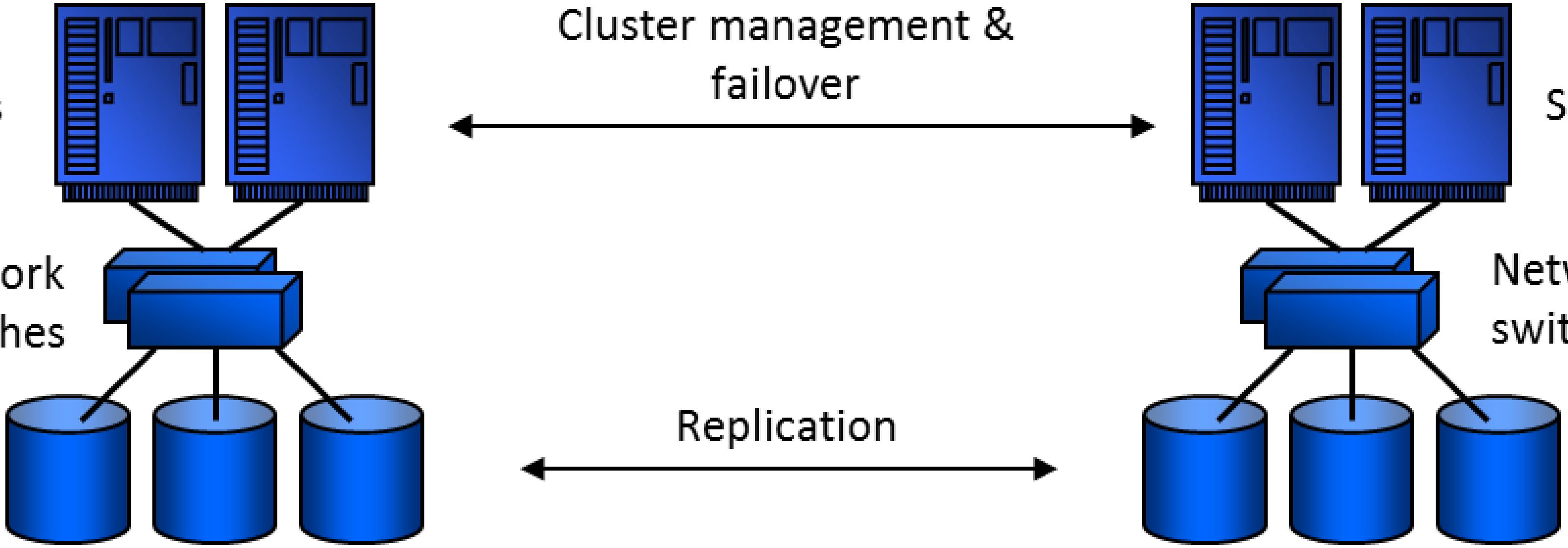
Data Availability

- Approaches (contd.)
  - replication and mirroring
    - mirroring is act of performing same write operations on two or more identical disks simultaneously (always synchronous, e.g. RAID level 1)
    - replication is act of propagating data written to one device over a network onto another device (synchronous, semi-synchronous, or asynchronous)
    - synchronous replication and mirroring provide near real time redundant copies of data

Data Availability

- Approaches (contd.)
  - Disaster tolerance
    - to guarantee a tight RPO and RTO under any circumstances, remote data replication is needed using, e.g., a WAN
    - asynchronous replication less sensitive to network latency
    - remote site should be fully operational including up to date data and DBMS functionality
    - in some implementations, both primary and backup DBMS are conceived as nodes in cluster that spans both the primary and remote data center (stretched cluster)

Data Availability



Servers

Cluster management & failover

Servers

Network switches

Network switches

Replication

Primary site

Remote site

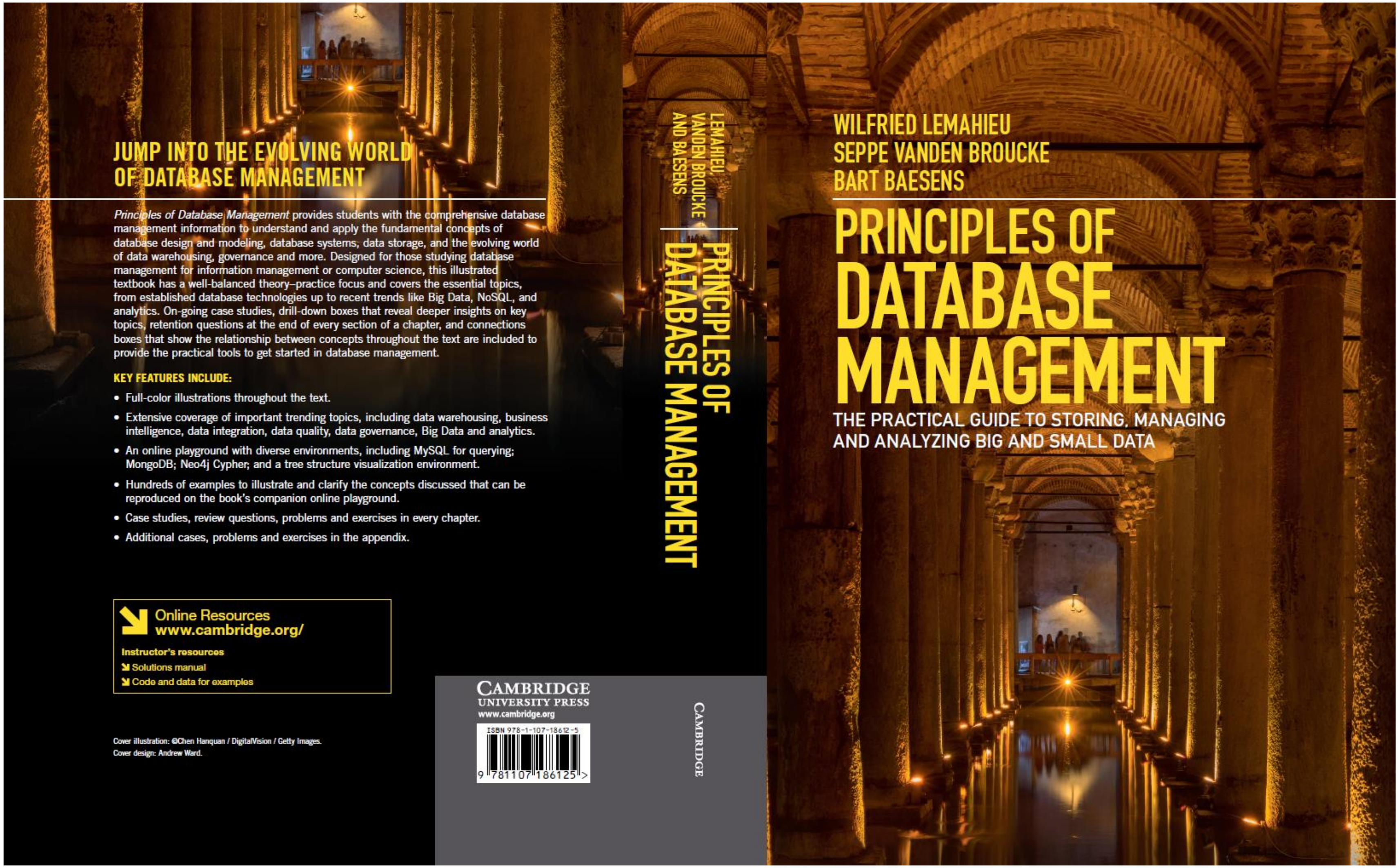
Data Availability

- Approaches (contd.)
  - Transaction recovery
    - transaction context must be preserved
    - if overall data replication is coordinated at DBMS level, then typically transaction context is also transferred between DBMSs
    - example is log shipping which means that logfile is replicated between both DBMSs
    - remote DBMS can use this log file for transaction recovery

Conclusions

- Physical Database Organization and Database Access Methods
- Enterprise Storage Subsystems and Business Continuity

More information?



JUMP INTO THE EVOLVING WORLD OF DATABASE MANAGEMENT

Principles of Database Management provides students with the comprehensive database management information to understand and apply the fundamental concepts of database design and modeling, database systems, data storage, and the evolving world of data warehousing, governance and more. Designed for those studying database management for information management or computer science, this illustrated textbook has a well-balanced theory–practice focus and covers the essential topics, from established database technologies up to recent trends like Big Data, NoSQL, and analytics. On-going case studies, drill-down boxes that reveal deeper insights on key topics, retention questions at the end of every section of a chapter, and connections boxes that show the relationship between concepts throughout the text are included to provide the practical tools to get started in database management.

KEY FEATURES INCLUDE:

- Full-color illustrations throughout the text.
- Extensive coverage of important trending topics, including data warehousing, business intelligence, data integration, data quality, data governance, Big Data and analytics.
- An online playground with diverse environments, including MySQL for querying; MongoDB; Neo4j Cypher; and a tree structure visualization environment.
- Hundreds of examples to illustrate and clarify the concepts discussed that can be reproduced on the book's companion online playground.
- Case studies, review questions, problems and exercises in every chapter.
- Additional cases, problems and exercises in the appendix.

Online Resources
www.cambridge.org/
Instructor's resources
Solutions manual
Code and data for examples

Cover illustration: ©Chen Hanquan / DigitalVision / Getty Images.
Cover design: Andrew Ward.

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

ISBN 978-1-107-18612-5

LEMAHIEU
VANDEN BROUCKE
AND BAESENS

PRINCIPLES OF
DATABASE MANAGEMENT

CAMBRIDGE

WILFRIED LEMAHIEU
SEPPE VANDEN BROUCKE
BART BAESENS

PRINCIPLES OF
DATABASE
MANAGEMENT

THE PRACTICAL GUIDE TO STORING, MANAGING
AND ANALYZING BIG AND SMALL DATA

www.pdbmbook.com