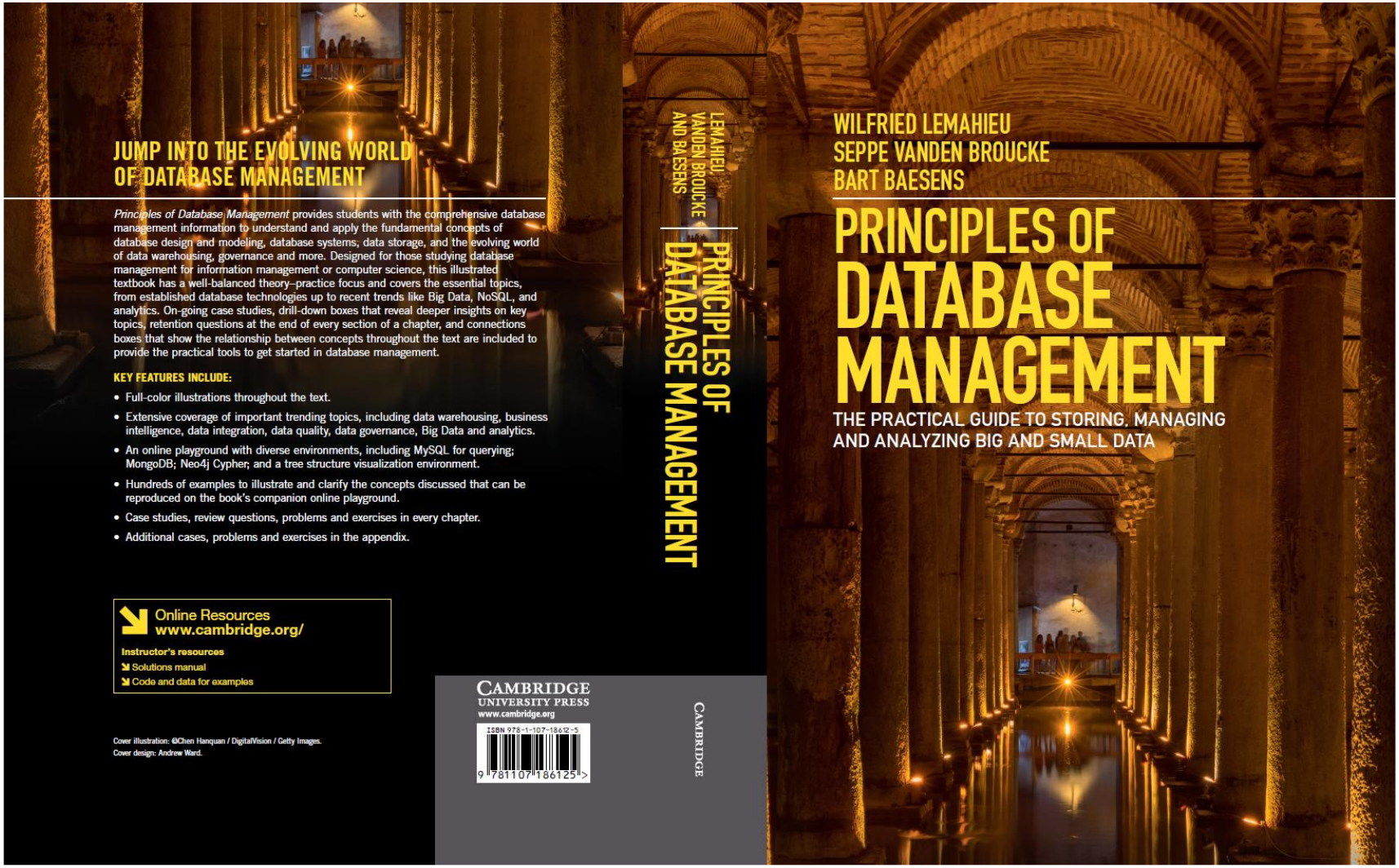
Architecture and Classification of DBMSs

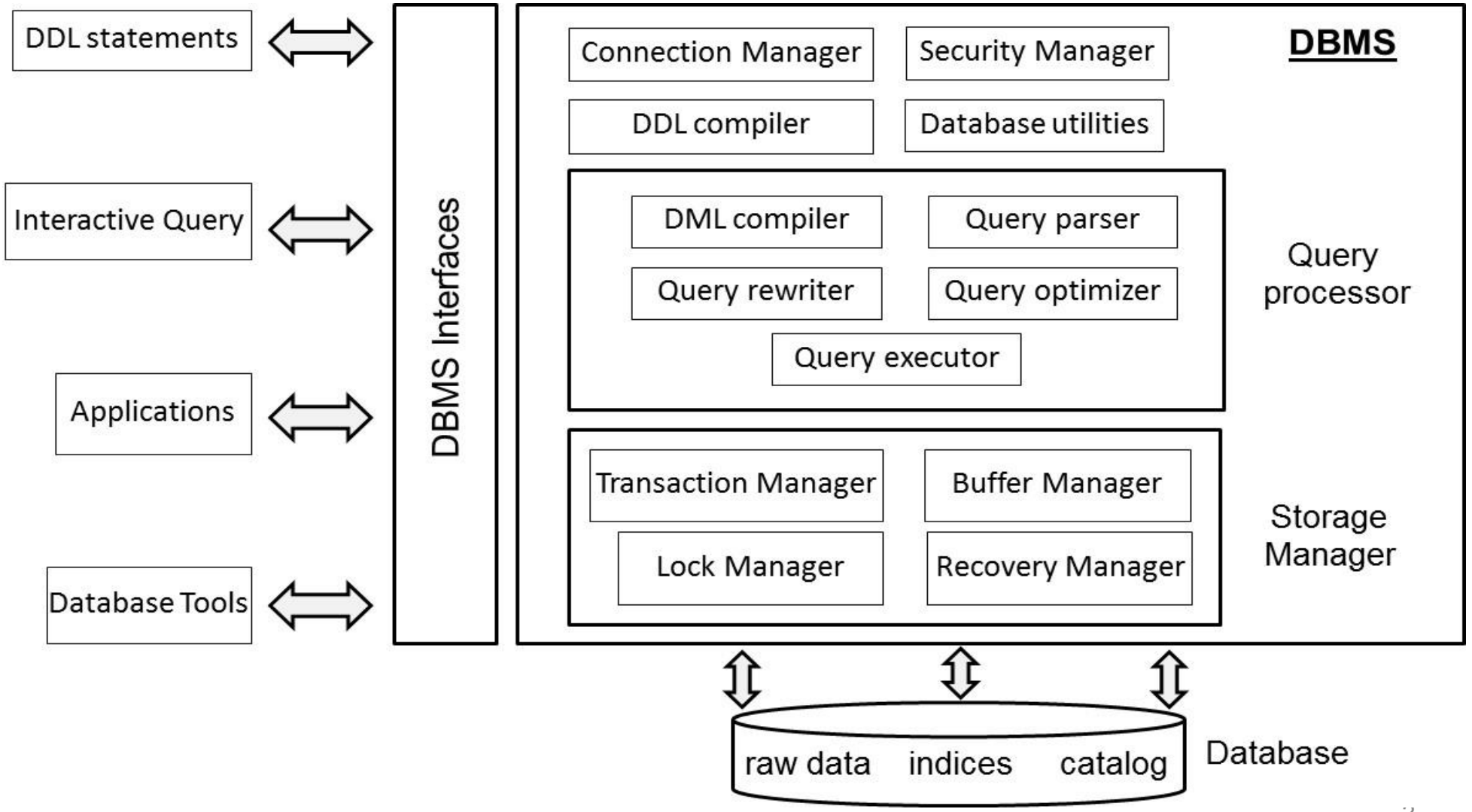


www.pdbmbook.com

Introduction

- Architecture of a DBMS
- Categorization of DBMSs

Architecture of a DBMS



Architecture of a DBMS

- Connection and Security Manager
- DDL Compiler
- Query Processor
- Storage Manager
- DBMS Utilities
- DBMS Interfaces

Connection and Security Manager

- Connection manager provides facilities to setup a database connection (locally or through a network)
  - verifies logon credentials and returns a connection handle
  - database connection can either run as single process or as thread within a process
- Security manager verifies whether a user has the right privileges
  - read versus write access

DDL Compiler

- Compiles the data definitions specified in DDL
- Ideally 3 DDLs (internal/logical/external data model)
- DDL compiler first parses the DDL definitions and checks their syntactical correctness
- DDL compiler then translates the data definitions to an internal format and generates errors if required
- Upon successful compilation, DDL compiler registers the data definitions in the catalog

Query processor

- Query processor assists in the execution of database queries such as retrieval, insertion, update or removal of data
- Key components:
  - DML compiler
  - Query parser
  - Query rewriter
  - Query optimizer
  - Query executor

DML Compiler

- DML compiler compiles the DML statements
- Procedural DML
  - DML explicitly specifies how to navigate in the database
  - record-at-a-time DML
  - no query processor
- Declarative DML
  - DML specifies what data should be retrieved or what changes should be made
  - set-at-a-time DML
  - query processor

DML Compiler

```
import java.sql.*;
public class JDBCExample1 {
public static void main(String[] args) {
try {
System.out.println("Loading JDBC driver...");
Class.forName("com.mysql.jdbc.Driver");
System.out.println("JDBC driver loaded!");
} catch (ClassNotFoundException e) {
throw new RuntimeException(e);
}
String url =
"jdbc:mysql://localhost:3306/employeeschema";
String username = "root";
String password = "mypassword123";
String query = "select E.Name, D.DName" +
"from employee E, department D" +
"where E.DNR=D.DNR;";
Connection connection = null;
Statement stmt=null;
```

```
try {
System.out.println("Connecting to database");
connection = DriverManager.getConnection(url,
username, password);
System.out.println("MySQL Database connected!");
stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
System.out.print(rs.getString(1));
System.out.print(" ");
System.out.println(rs.getString(2));
}
stmt.close();
} catch (SQLException e) {
System.out.println(e.toString());
} finally {
System.out.println("Closing the connection.");
if (connection != null) {
try {
connection.close();
} catch (SQLException ignore) {}}}}
```

DML Compiler

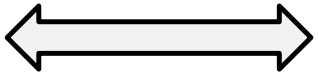
- Impedance mismatch problem
  - mapping between OO (e.g. Java) and relational (e.g. SQL) concepts
- Impedance mismatch solutions
  - host language and DBMS with comparable data structures (e.g., Java and OODBMS)
  - middleware to map the data structures from the DBMS to the host language and vice versa

DML Compiler

Java

```
public class Employee {
private int EmployeeID;
private String Name;
private String Gender;
private int DNR;

public int getEmployeeID() {
return EmployeeID;
}
public void setEmployeeID( int id ) {
this.EmployeeID = id;
}
public String getName() {
return Name;
}
public void setName( String name ) {
this.Name = name;
}
…}
```



SQL

```
CREATE TABLE Employee (
'EmployeeID' INT NOT NULL,
'Name'  VARCHAR(45) NULL,
'Gender' VARCHAR(45) NULL,
'DNR' INT NULL)
```

EmployeeID	Name	Gender	DNR
100	Bart Baesens	Male	2
110	Wilfried Lemahieu	Male	4
120	Seppe vanden Broucke	Male	6
…			

DML Compiler

- DML compiler starts by extracting the DML statements from the host language.
- DML compiler then collaborates with the query parser, query rewriter, query optimizer, and query executor for executing the DML statements
- Errors are generated and reported if necessary

Query Parser and Query Rewriter

- Query parser parses the query into an internal representation format
- Query parser checks the query for syntactical and semantical correctness
- Query rewriter optimizes the query, independently of the current database state

Query Optimizer

- Query optimizer optimizes the query based upon the current database state (based upon e.g. predefined indexes)
- Query optimizer comes up with various query execution plans and evaluates their cost in terms of estimated
  - number of I/O operations
  - CPU processing cost
  - execution time
- Estimates based on catalog information combined with statistical inference
- Query optimizer is a key competitive asset of a DBMS

Query executor

- Result of the query optimization is a final execution plan
- Query executor takes care of the actual execution by calling on the storage manager to retrieve the data requested

Storage manager

- Storage manager governs physical file access and supervises the correct and efficient storage of data
- Storage manager consists of
  - transaction manager
  - buffer manager
  - lock manager
  - recovery manager

Transaction manager

- Transaction manager supervises execution of database transactions
  - a database transaction is a sequence of read/write operations considered to be an atomic unit
- Transaction manager creates a schedule with interleaved read/write operations
- Transaction manager guarantees ACID properties
- COMMIT a transaction upon successful execution and ROLLBACK a transaction upon unsuccessful execution

Buffer Manager

- Buffer manager manages buffer memory of the DBMS
- Buffer manager intelligently caches data in the buffer
- Example strategies:
  - Data locality: data recently retrieved is likely to be retrieved again
  - 20/80 law: 80% of the transactions read or write only 20% of the data
- Buffer manager needs to adopt smart replacement strategy in case buffer is full
- Buffer manager needs to interact with lock manager

Lock Manager

- Lock manager provides concurrency control which ensures data integrity at all times
- Two types of locks: read and write locks
- Lock manager is responsible for assigning, releasing, and recording locks in the catalog
- Lock manager makes use of a *locking protocol* which describes the locking rules, and a lock table with the lock information

Recovery Manager

- Recovery manager supervises the correct execution of database transactions
- Recovery manager keeps track of all database operations in a log file
- Recovery manager will be called upon to undo actions of aborted transactions or during crash recovery

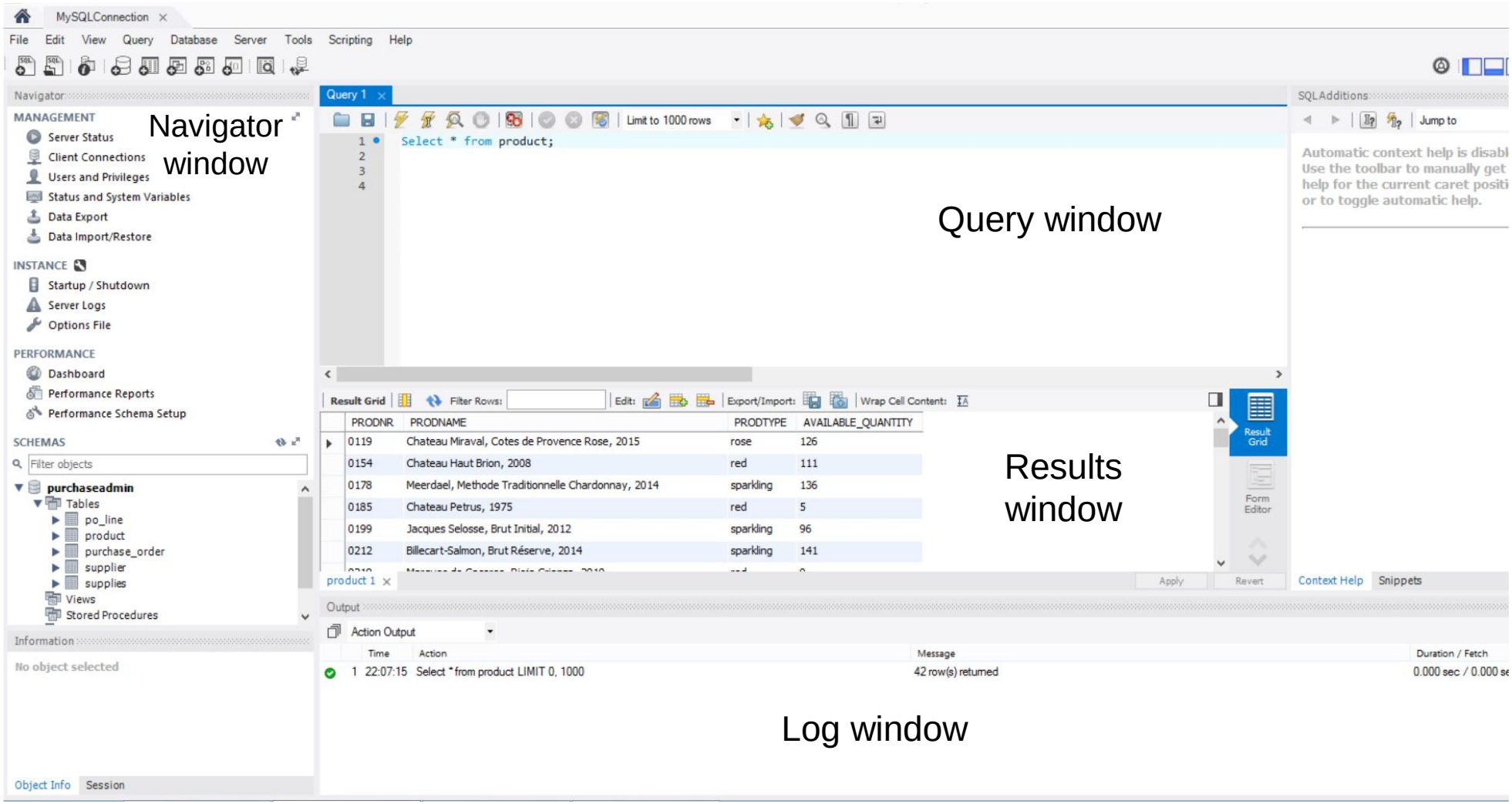
DBMS Utilities

- Loading utility
- Reorganization utility
- Performance monitoring utilities
- User management utilities
- Backup and recovery utility

DBMS Interfaces

- Web-based interface
- Stand-alone query language interface
- Command line interface
- Forms-based interface
- Graphical user interface
- Natural language interface
- Admin interface
- Network interface
- …

DBMS Interfaces



Categorization of DBMSs

- Categorization based on data model
- Categorization based on degree of simultaneous access
- Categorization based on architecture
- Categorization based on usage

Categorization based on data model

- Hierarchical DBMSs
  - adopt a tree like data model
  - DML is procedural and record oriented
  - no query processor (logical and internal data model intertwined)
  - E.g., IMS (IBM)
- Network DBMSs
  - use a network data model
  - CODASYL DBMSs
  - DML is procedural and record oriented
  - no query processor (logical and internal data model intertwined)
  - CA-IDMS (Computer Associates)

Categorization based on data model

- Relational DBMSs
  - use the relational data model
  - currently the most popular in industry
  - SQL (declarative and set oriented)
  - query processor
  - strict separation between the logical and internal data model
  - E.g., MySQL (open source, Oracle), Oracle DBMS (Oracle), DB2 (IBM), Microsoft SQL (Microsoft)

Categorization based on data model

- Object-Oriented DBMSs (OODBMS)
  - based upon the OO data model
  - No impedance mismatch in combination with OO host language
  - E.g., db4o (open source, Versant), Caché (Intersystems) GemStone/S (GemTalk Systems)
  - only successfull in niche markets, due to their complexity

Categorization based on data model

- Object-Relational DBMSs (ORDBMSs)
  - also referred to as extended relational DBMSs (ERDBMSs)
  - use a relational model extended with OO concepts
  - DML is SQL (declarative and set oriented)
  - E.g., Oracle DBMS (Oracle), DB2 (IBM), Microsoft SQL (Microsoft)

Categorization based on data model

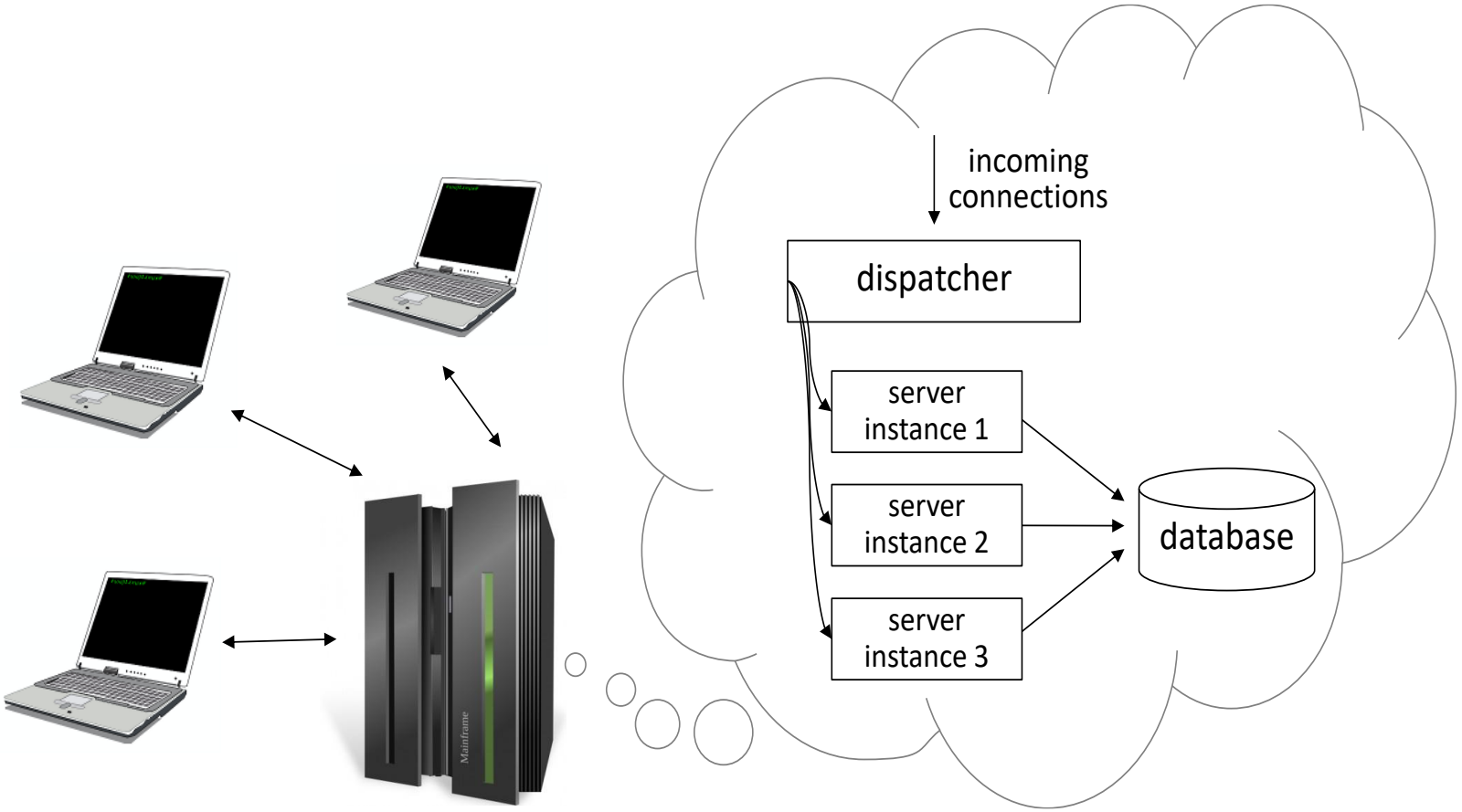
- XML DBMSs
  - use the XML data model to store data
  - Native XML DBMSs (e.g., BaseX, eXist) map the tree structure of an XML document to a physical storage structure
  - XML-enabled DBMSs (e.g., Oracle, IBM DB2) are existing DBMSs that are extended with facilities to store XML data

Categorization based on data model

- NoSQL DBMSs
  - targeted at storing big and unstructured data
  - can be classified into key-value stores, column-oriented databases and graph databases
  - focus on scalability and the ability to cope with irregular or highly volatile data structures
  - E.g., Apache Hadoop, MongoDB, Neo4j

Categorization based upon degree of simultaneous access

- Single user versus multi user systems



Categorization based on architecture

- Centralized DBMS architecture
  - data is maintained on a centralized server
- Client server DBMS architecture
  - active clients request services from passive servers
  - fat server versus fat client variant
- n-tier DBMS architecture
  - client with GUI functionality, application server with applications, database server with DBMS and database, and web server for web based access

Categorization based on architecture

- Cloud DBMS architecture
  - DBMS and database are hosted by a third-party cloud provider
  - E.g., Apache Cassandra project and Google's BigTable
- Federated DBMS
  - provides a uniform interface to multiple underlying data sources
  - hides the underlying storage details to facilitate data access

Categorization based on architecture

- in-memory DBMS
  - stores all data in internal memory instead of slower external storage (e.g., disk)
  - often used for real-time purposes
  - E.g., HANA (SAP)

Categorization based on usage

- On-line transaction processing (OLTP)
  - focus on managing operational or transactional data
  - database server must be able to process lots of simple transactions per unit of time
  - DBMS must have good support for processing a high volume of short, simple queries
- On-line analytical processing (OLAP)
  - focus on using operational data for tactical or strategical decision making
  - limited number of users formulates complex queries
  - DBMS should support efficient processing of complex queries which often come in smaller volumes

Categorization based on usage

- Big Data & Analytics
  - NoSQL databases
  - focus on more flexible, or even schema-less, database structures
  - store unstructured information such as emails, text documents, Twitter tweets, Facebook posts, etc.
- Multimedia
  - Multimedia DBMSs provide storage of multimedia data such as text, images, audio, video, 3D games, etc.
  - should also provide content-based query facilities

Categorization Based on Usage

- Spatial applications
  - spatial DBMSs support storage and querying of spatial data (both 2D and 3D)
  - Geographical Information Systems (GIS)
- Sensoring
  - sensor DBMSs manage sensor data such as biometric data from wearables, or telematics data

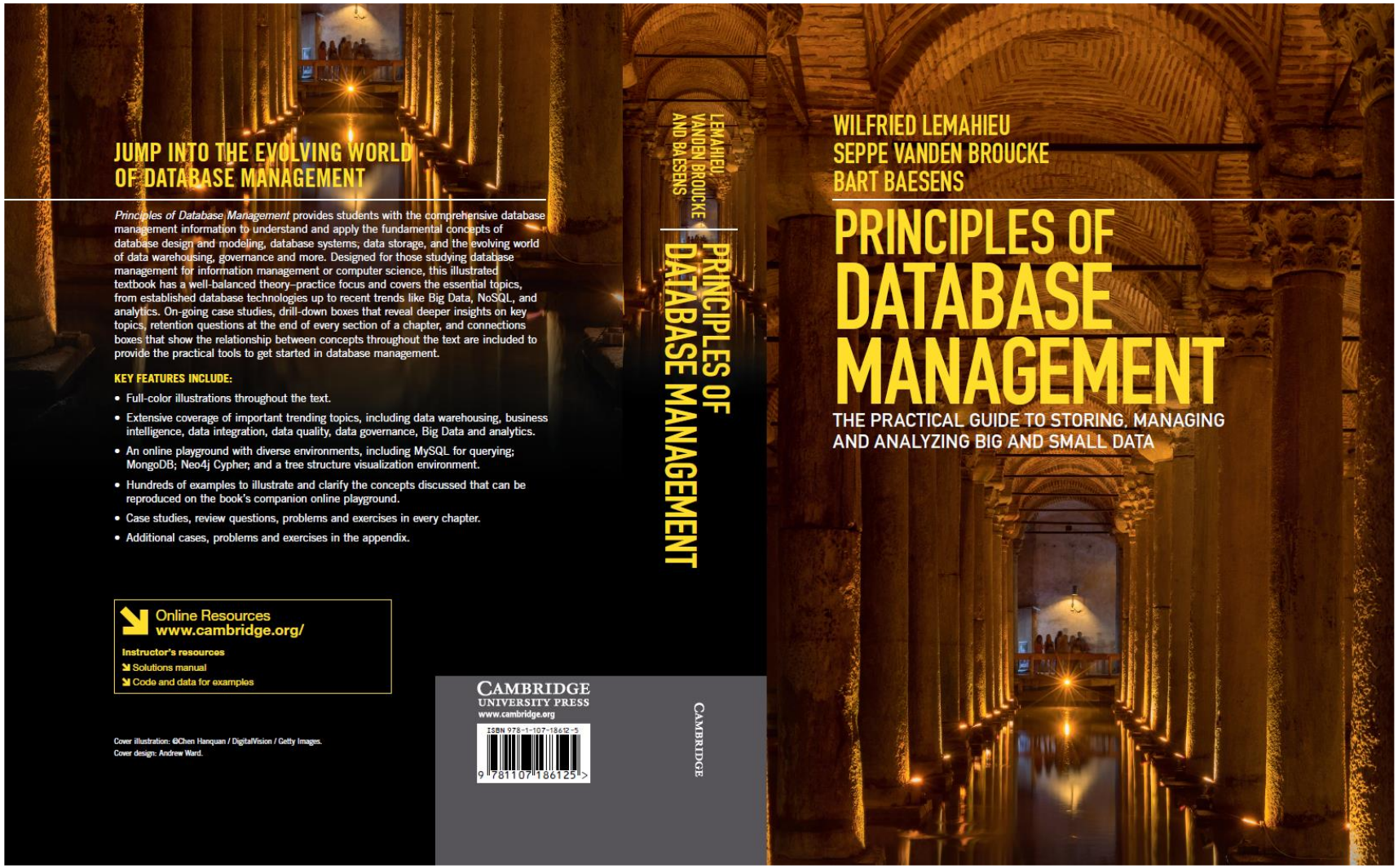
Categorization based on usage

- Mobile
  - Mobile DBMSs run on smartphones, tablets or other mobile devices.
  - should always be online, have a small footprint and be able to deal with limited processing power, storage and battery life
- Open source
  - code of open source DBMSs is publicly available and can be extended by anyone
  - See www.sourceforge.net
  - E.g., MySQL (Oracle)

Conclusions

- Architecture of a DBMS
- Categorization of DBMSs

More information?



www.pdbmbook.com