Database Systems

Session 1 – Sub-Topic 1
DBMSs Summarized

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

1. Session Overview
2. Fundamental Concepts of Database Management
3. Architecture and Classification of DBMSs
4. Organizational Aspects of Data Management
5. Summary and Conclusion

Agenda

- Fundamental Concepts of Database Management
  - Applications of Database Technology
  - Key definitions
  - File versus Database Approach to Data Management
  - Elements of a Database System
  - Advantages of Database Systems and Database Management
- Architecture and Classification of DBMSs
  - Architecture of a DBMS
  - Categorization of DBMSs
- Organizational Aspects of Data Management
  - Data Management
  - Roles in Data Management

Icons / Metaphors

Information

Common Realization

Knowledge/Competency Pattern

Governance

Alignment

Solution Approach

Agenda

1. Session Overview
2. Fundamental Concepts of Database Management
3. Architecture and Classification of DBMSs
4. Organizational Aspects of Data Management
5. Summary and Conclusion

Agenda

- Fundamental Concepts of Database Management
  - Applications of Database Technology
  - Key definitions
  - File versus Database Approach to Data Management
  - Elements of a Database System
  - Advantages of Database Systems and Database Management

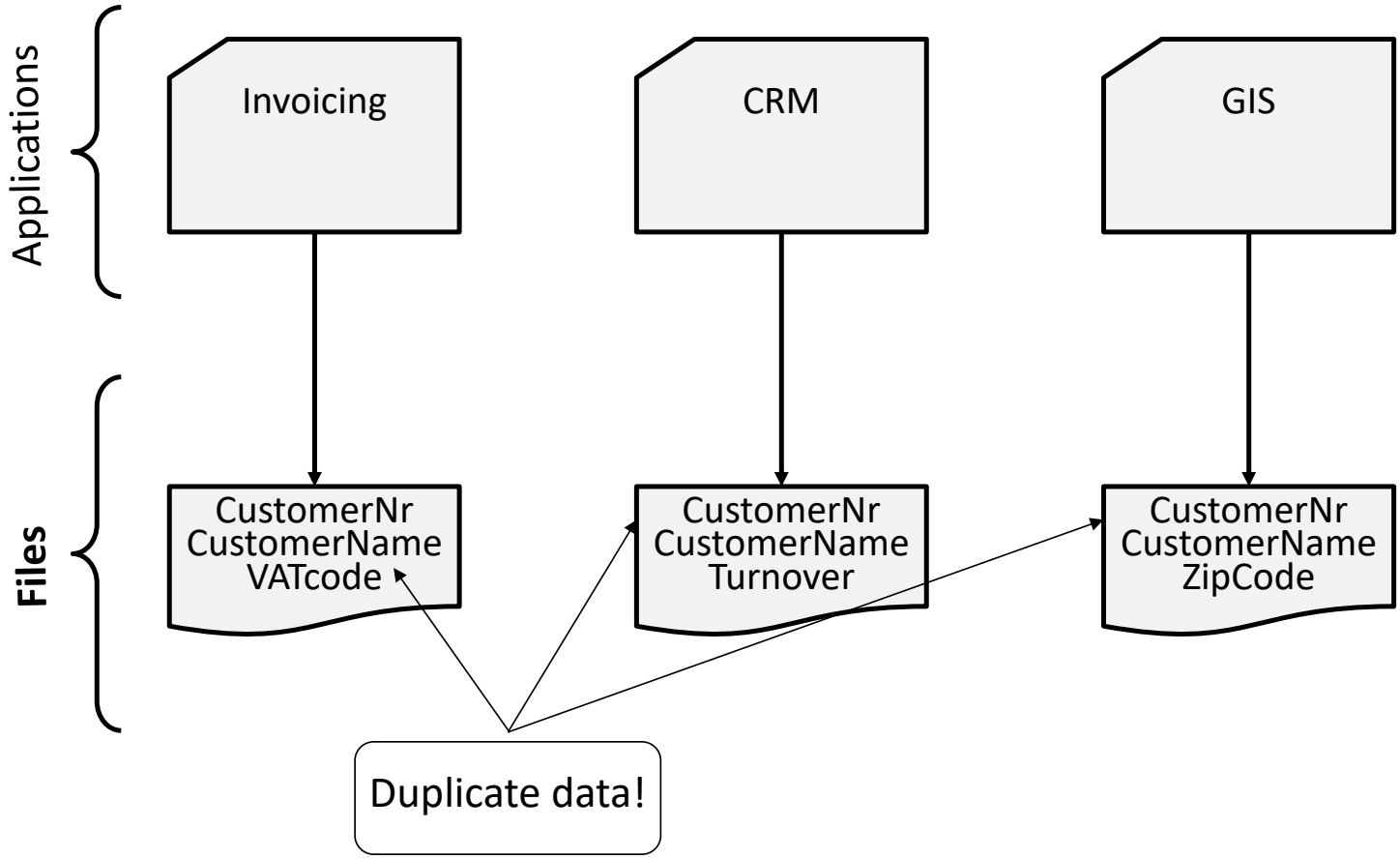
Applications of Database Technology

- Storage and retrieval of traditional numeric and alphanumeric data in an inventory application
- Multimedia applications (e.g., YouTube, Spotify)
- Biometric applications (e.g., fingerprints, retina scans)
- Wearable applications (e.g., FitBit, Apple Watch)
- Geographical Information Systems (GIS) applications (e.g., Google Maps)
- Sensor applications (e.g., nuclear reactor)
- Big Data applications (e.g., Walmart)
- Internet of Things (IoT) applications (e.g., Telematics)

Key definitions

- A database can be defined as a collection of related data items within a specific business process or problem setting
  - » has a target group of users and applications
- A Database Management System (DBMS), is the software package used to define, create, use and maintain a database
  - » consists of several software modules
- The combination of a DBMS and a database is then often called a database system

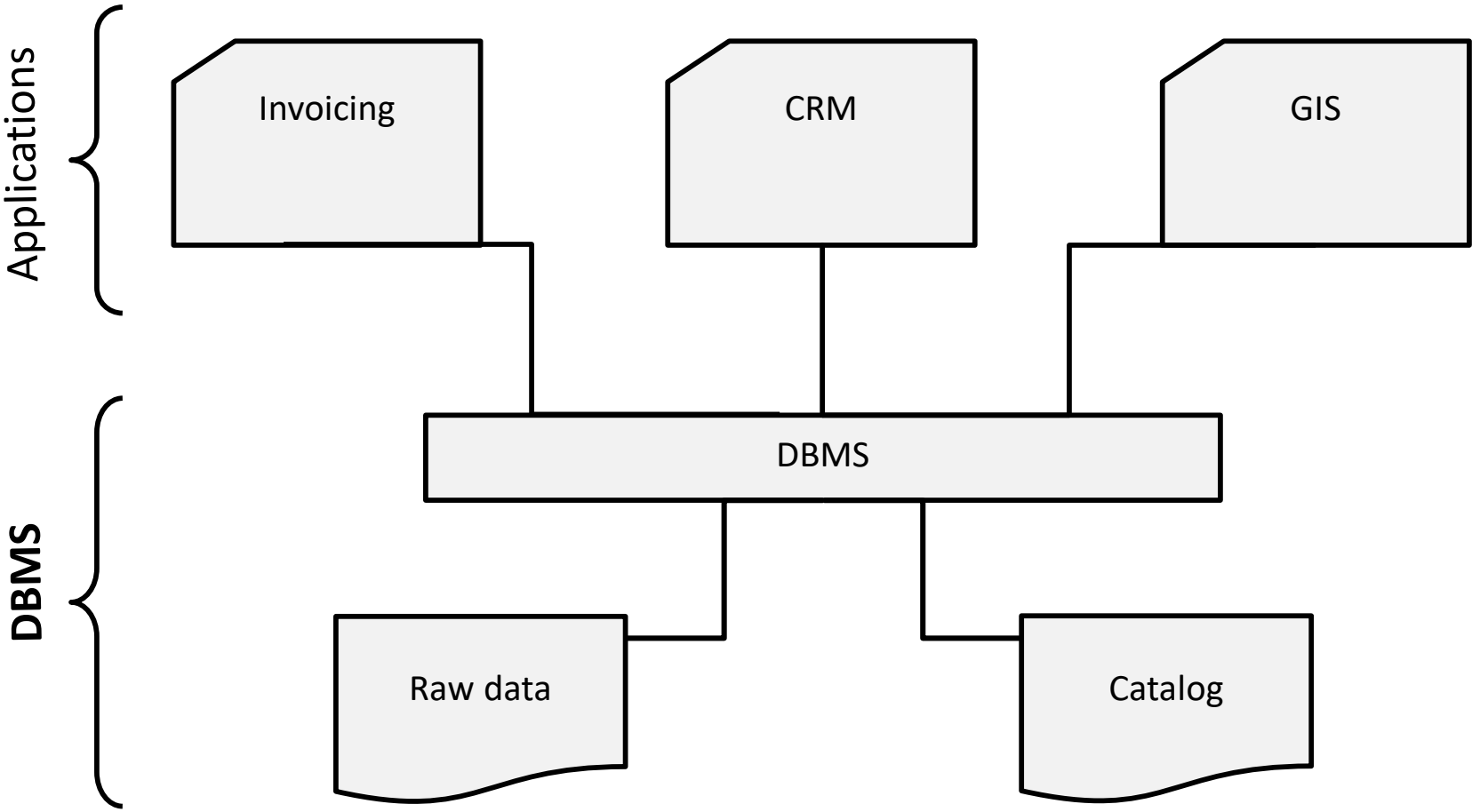
File versus Database Approach to Data Management



File versus Database Approach to Data Management

- File approach
  - duplicate or redundant information will be stored
  - danger of inconsistent data
  - strong coupling between applications and data
  - hard to manage concurrency control
  - hard to integrate applications aimed at providing cross-company services

File versus Database Approach to Data Management



File versus Database Approach to Data Management

- Database approach
  - superior to the file approach in terms of efficiency, consistency and maintenance
  - loose coupling between applications and data
  - facilities provided for data querying and retrieval

File versus Database Approach to Data Management

- File approach

```
Procedure FindCustomer;
begin
     open file Customer.txt;
Read(Customer)
While not EOF(Customer)
If Customer.name='Bart' then
display(Customer);
EndIf
Read(Customer);
EndWhile;
End;
```

- Database approach (SQL)

```
SELECT *
FROM Customer
WHERE
name = 'Bart'
```

Elements of a Database System

- Database model versus instances
- Data Model
- The Three Layer/Schema Architecture
- Catalog
- Database Users
- Database Languages

Database model versus instances

- Database model or database schema provides the description of the database data at different levels of detail and specifies the various data items, their characteristics and relationships, constraints, storage details, etc.
  - specified during database design and not expected to change too frequently
  - stored in the catalog
- Database state represents the data in the database at a particular moment
  - also called the current set of instance
  - typically changes on an ongoing basis

Database model versus instances

- Database model

  Student (number, name, address, email)

  Course (number, name)

  Building (number, address)

- Database state

STUDENT			
Number	**Name**	**Address**	**Email**
0165854	Bart Baesens	1040 Market Street, SF	Bart.Baesens@kuleuven.be
0168975	Seppe vanden Broucke	520, Fifth Avenue, NY	Seppe.vandenbroucke@kuleuven.be
0157895	Wilfried Lemahieu	644, Wacker Drive, Chicago	Wilfried.Lemahieu@kuleuven.be

COURSE	
Number	**Name**
D0I69A	Principles of Database Management
D0R04A	Basic Programming
D0T21A	Big Data & Analytics

BUILDING	
Number	**Address**
0600	Naamsestraat 69, Leuven
0365	Naamsestraat 78, Leuven
0589	Tiensestraat 115, Leuven

Data Model

- A database model is comprised of different data models, each describing the data from different perspectives
- A data model provides a clear and unambiguous description of the data items, their relationships and various data constraints from a particular perspective

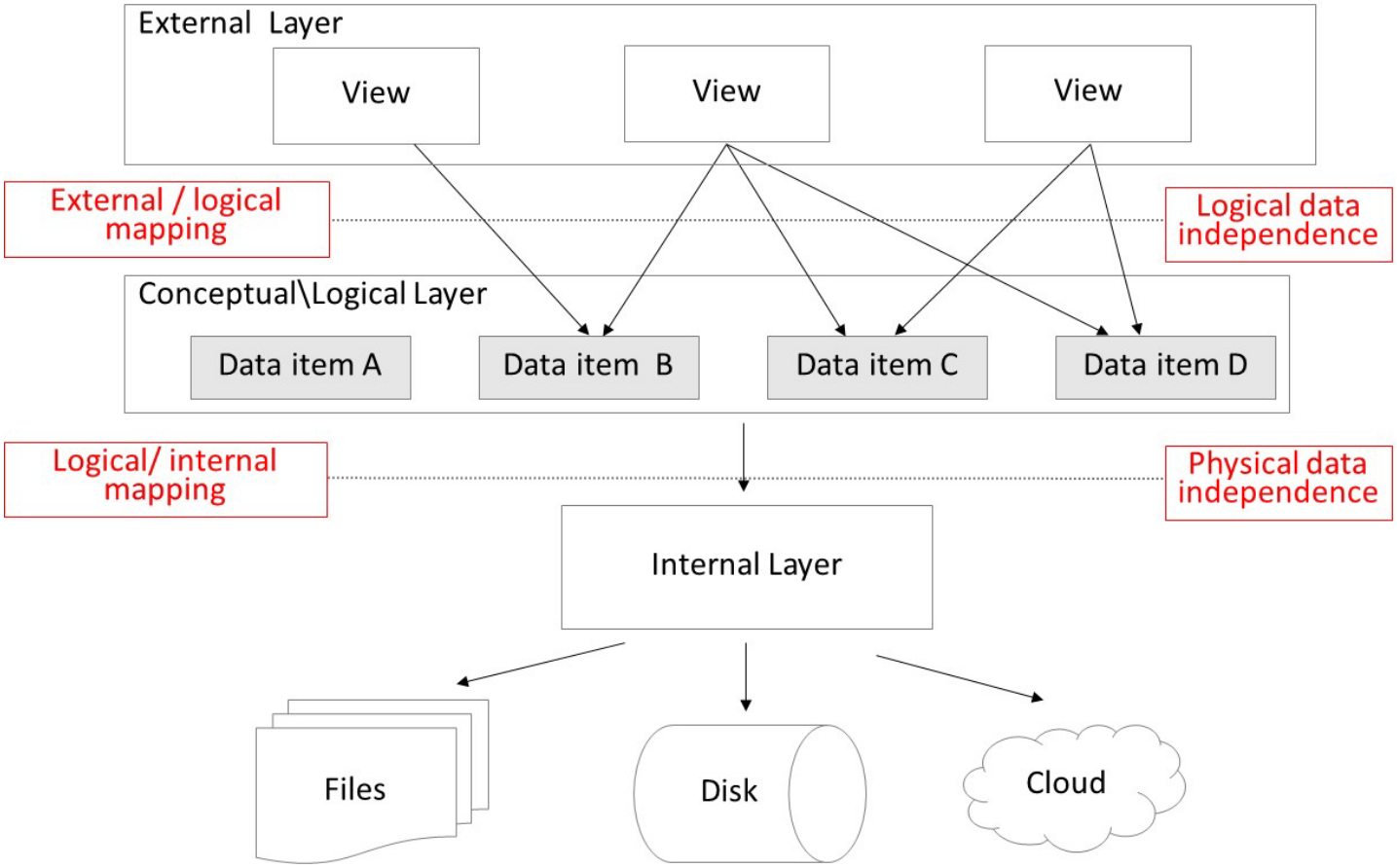
Data Model

- A conceptual data model provides a high-level description of the data items with their characteristics and relationships
  - communication instrument between information architect and business user
  - should be implementation independent, user-friendly, and close to how the business user perceives the data
  - usually represented using an Enhanced-Entity Relationship (EER) model, or an object-oriented model
- Logical data model is a translation or mapping of the conceptual data model towards a specific implementation environment
  - can be a hierarchical, CODASYL, relational, object-oriented, extended relational, XML or NoSQL model, or NewSQL

Data Model

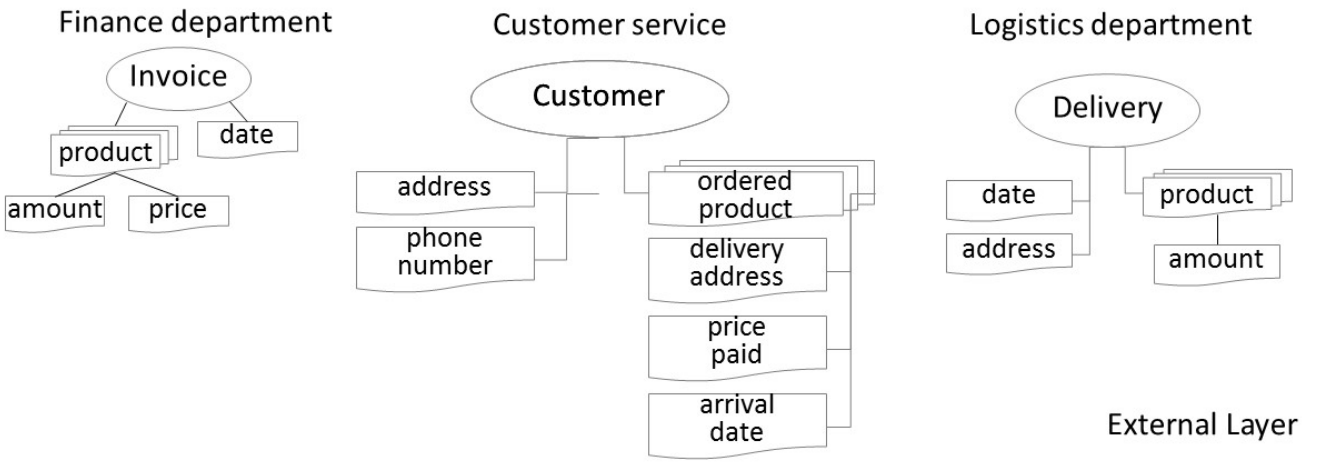
- Logical data model can be mapped to an internal data model that represents the data's physical storage details
  - clearly describes which data is stored where, in what format, which indexes are provided to speed up retrieval, etc.
  - highly DBMS specific
- External data model contains various subsets of the data items in the logical model, also called views, tailored towards the needs of specific applications or groups of users

The Three Layer/Schema Architecture



physical data + logical data independence!

The Three Layer Architecture



Catalog

- Heart of the DBMS
- Contains the data definitions, or metadata, of your database application
- Stores the definitions of the views, logical and internal data models, and synchronizes these three data models to make sure their consistency is guaranteed

Database Users

- Information architect designs the conceptual data model
  - closely interacts with the business user
- Database designer translates the conceptual data model into a logical and internal data model
- Database administrator (DBA) is responsible for the implementation and monitoring of the database
- Application developer develops database applications in a programming language such as Java or Python
- Business user will run these applications to perform specific database operations

Database Languages

- Data Definition Language (DDL) is used by the DBA to express the database's external, logical and internal data models
  - definitions are stored in the catalog
- Data Manipulation Language (DML) is used to retrieve, insert, delete, and modify data
  - DML statements can be embedded in a programming language, or entered interactively through a front-end querying tool
- Structured Query Language (SQL) offers both DDL and DML statements for relational database systems

Advantages of Database Systems and Database Management

- Data Independence
- Database Modeling
- Managing Structured, Semi-Structured and Unstructured Data
- Managing Data Redundancy
- Specifying Integrity Rules
- Concurrency Control
- Backup and Recovery Facilities
- Data Security
- Performance Utilities

Data Independence

- Data independence implies that changes in data definitions have minimal to no impact on the applications
- Physical data independence implies that neither the applications, nor the views or logical data model must be changed when changes are made to the data storage specifications in the internal data model
  - » DBMS should provide interfaces between logical and internal data models
- Logical data independence implies that software applications are minimally affected by changes in the conceptual or logical data model
  - » views in the external data model will act as a protective shield
  - » DBMS must provide interfaces between conceptual/logical and external layer

Database Modeling

- A data model is an explicit representation of the data items together with their characteristics and relationships
- A conceptual data model should provide a formal and perfect mapping of the data requirements of the business process and is made in collaboration with the business user
  - » translated into logical and internal data model
- Important that a data model's assumptions and shortcomings are clearly documented

Managing Structured, Semi-Structured and Unstructured Data

- Structured data
  - can be described according to a formal logical data model
  - ability to express integrity rules and enforce correctness of data
  - also facilitates searching, processing and analyzing the data
  - E.g., number, name, address and email of a student
- Unstructured data
  - no finer grained components in a file or series of characters that can be interpreted in a meaningful way by a DBMS or application
  - E.g., document with biographies of famous NY citizens
  - Note: volume of unstructured data surpasses that of structured data

Managing Structured, Semi-Structured and Unstructured Data

- Semi-structured data
  - data which does have a certain structure, but the structure may be very irregular or highly volatile
  - E.g., individual users' webpages on a social media platform, or resume documents in a human resources database

Managing Data Redundancy

- Duplication of data can be desired in distributed environments to improve data retrieval performance
- DBMS is now responsible for the management of the redundancy by providing synchronization facilities to safeguard data consistency
- Compared to the file approach, the DBMS guarantees correctness of the data without user intervention

Specifying Integrity Rules

- Syntactical rules specify how the data should be represented and stored
  - » E.g., customerID is an integer; birthdate should be stored as month, day and year
- Semantical rules focus on the semantical correctness or meaning of the data
  - » E.g., customerID is unique; account balance should be > 0; customer cannot be deleted if he/she has pending invoices
- Integrity rules are specified as part of the conceptual\logical data model and stored in the catalog
  - » directly enforced by the DBMS instead of applications

Concurrency Control

- DBMS has built in facilities to support concurrent or parallel execution of database programs
- Key concept is a database transaction
  - sequence of read/write operations considered to be an atomic unit in the sense that either all operations are executed or none at all
- Read/write operations can be executed at the same time by the DBMS
- DBMS should avoid inconsistencies!

- Lost update problem

Time	T1	T2	balance
t1		Begin transaction	$100
t2	Begin transaction	read(balance)	$100
t3	read(balance)	balance=balance+120	$100
t4	balance=balance-50	write(balance)	$220
t5	write(balance)	End transaction	$50
t6	End transaction		$50

Concurrency Control

- DBMS must support ACID (Atomicity, Consistency, Isolation, Durability) properties
  - Atomicity requires that a transaction should either be executed in its entirety or not all
  - Consistency assures that a transaction brings the database from one consistent state to another
  - Isolation ensures that the effect of concurrent transactions should be the same as if they would have been executed in isolation
  - Durability ensures that the database changes made by a transaction declared successful can be made permanent under all circumstances

Backup and Recovery Facilities

- Backup and recovery facilities can be used to deal with the effect of loss of data due to hardware or network errors, or bugs in system or application software
- Backup facilities can either perform a full or incremental backup
- Recovery facilities allow to restore the data to a previous state after loss or damage occurred

Data Security

- Data security can be enforced by the DBMS
- Some users have read access, whilst others have write access to the data (role-based functionality)
  - E.g., vendor managed inventory (VMI)
- Data access can be managed via logins and passwords assigned to users or user accounts
- Each account has its own authorization rules that can be stored in the catalog

Performance Utilities

- Three KPIs of a DBMS are
  - response time denoting the time elapsed between issuing a database request and the successful termination thereof
  - throughput rate representing the transactions a DBMS can process per unit of time
  - space utilization referring to the space utilized by the DBMS to store both raw data and metadata
- DBMSs come with various types of utilities aimed at improving these KPIs
  - E.g., utilities to distribute and optimize data storage, to tune indexes for faster query execution, to tune queries to improve application performance, or to optimize buffer management

Summary

- Applications of Database Technology
- Key definitions
- File versus Database Approach to Data Management
- Elements of a Database System
- Advantages of Database Systems and Database Management

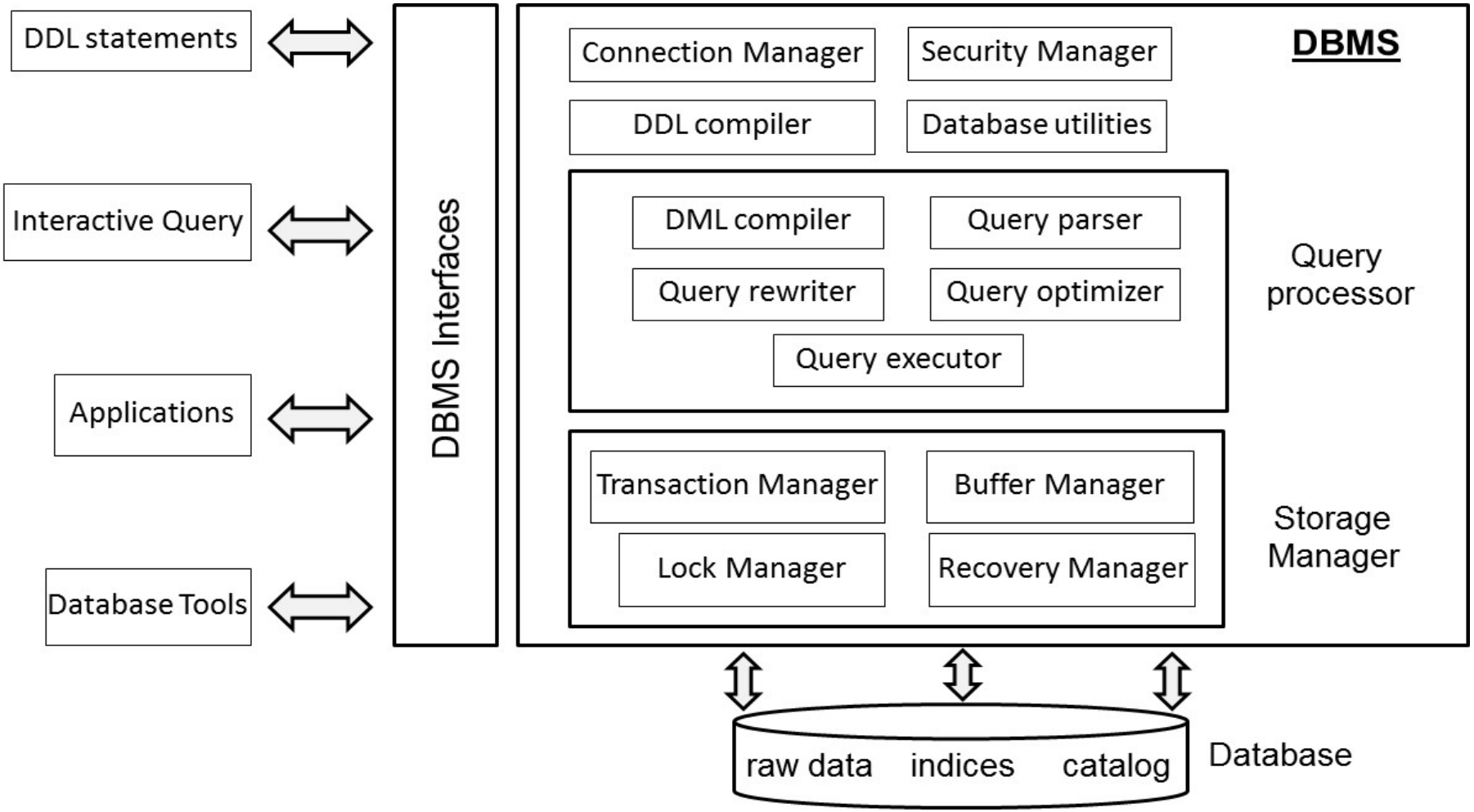
Agenda

1. Session Overview
2. Fundamental Concepts of Database Management
3. Architecture and Classification of DBMSs
4. Organizational Aspects of Data Management
5. Summary and Conclusion

Agenda

- Architecture and Classification of DBMSs
  - 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

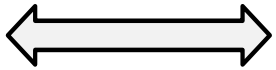
- Impedance mismatch problem
  - mapping between OO (e.g. Java) and relational (e.g. SQL) concepts
- Impedance mismatch solutions
  - host language and DBMS with comparable data structures (e.g., Java and OODBMS)
  - middleware to map the data structures from the DBMS to the host language and vice versa

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

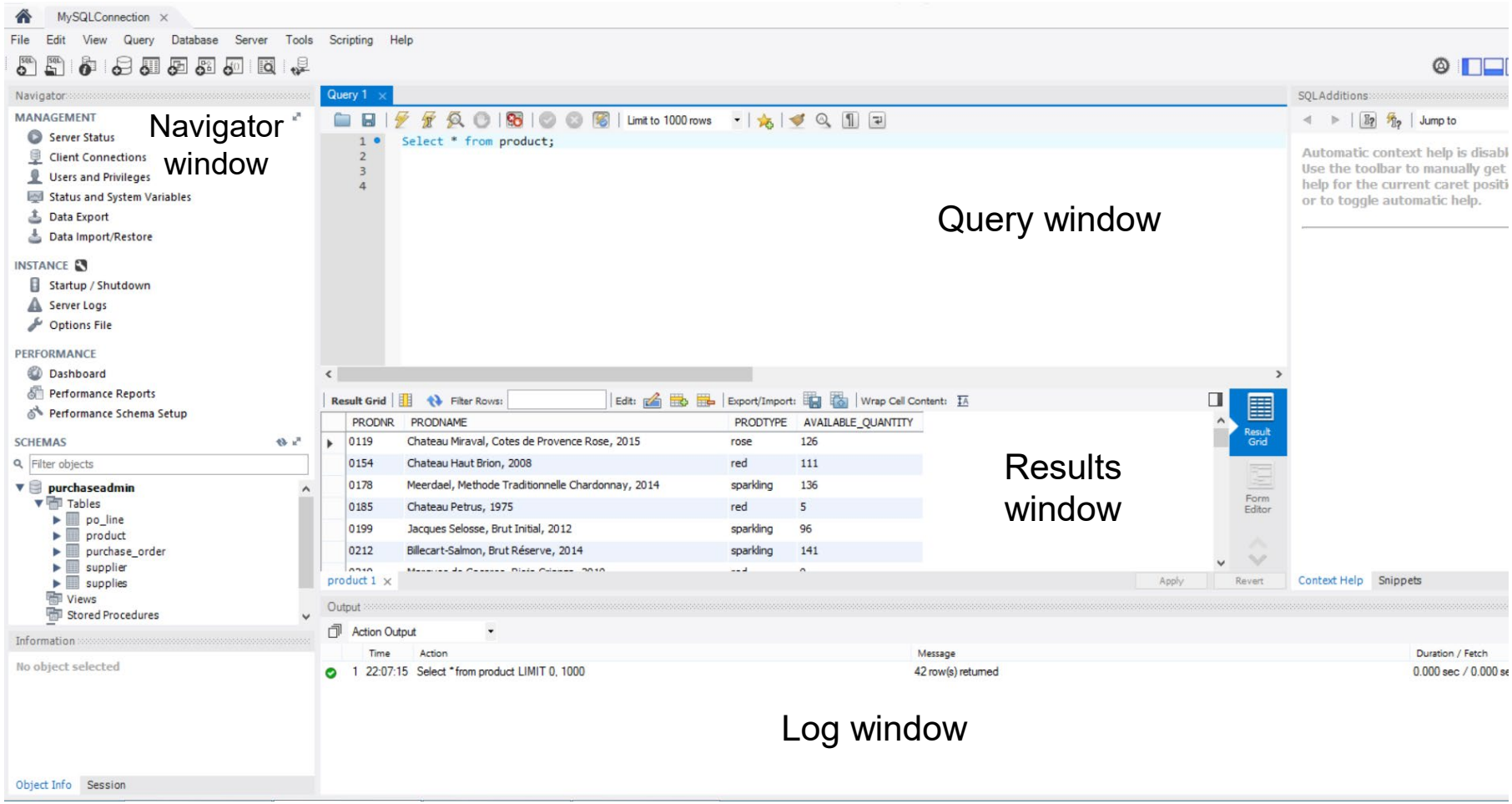
DBMS Utilities

- 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 (MySQL WorkBench)



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

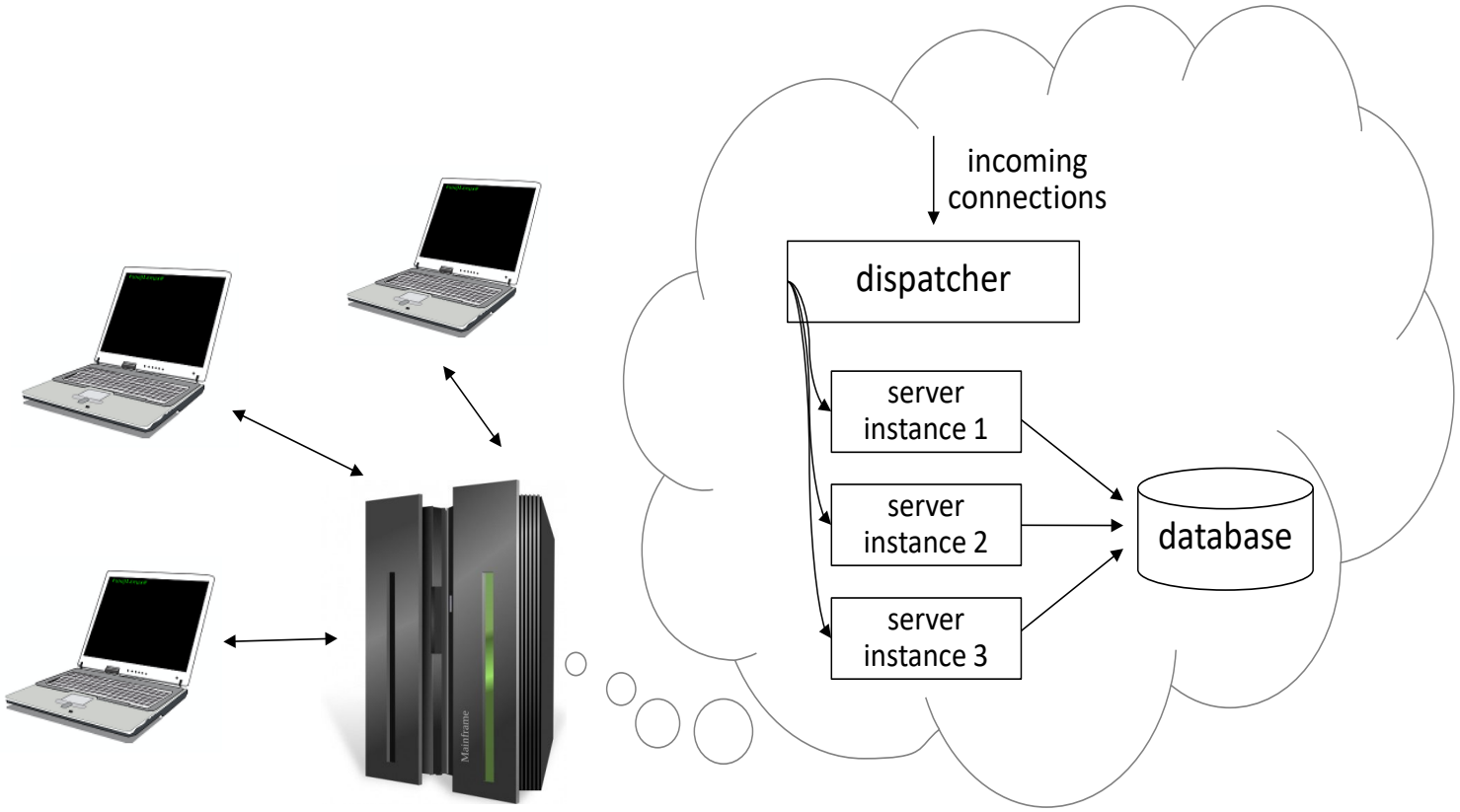
- XML/JSON DBMSs
  - use the XML data model and JSON to store data
  - Native XML DBMSs (e.g., BaseX, eXist) map the tree structure of an XML document to a physical storage structure
  - XML-enabled DBMSs (e.g., Oracle, IBM DB2) are existing DBMSs that are extended with facilities to store XML/JSON data

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)

Agenda

1. Session Overview
2. Fundamental Concepts of Database Management
3. Architecture and Classification of DBMSs
4. Organizational Aspects of Data Management
5. Summary and Conclusion

Agenda

- Organizational Aspects of Data Management
  - Data Management
  - Roles in Data Management

Data Management

- Catalogs and the Role of Metadata
- Metadata Modeling
- Data Quality
- Data Governance

Catalogs and the Role of Metadata

- Just as raw data, also metadata is data that needs to be properly modeled, stored and managed
- Concepts of data modeling should also be applied to metadata
- In a DBMS approach, metadata is stored in a catalog (a.k.a. data dictionary, data repository), which constitutes the heart of the database system
  - » can be part of a DBMS or standalone component

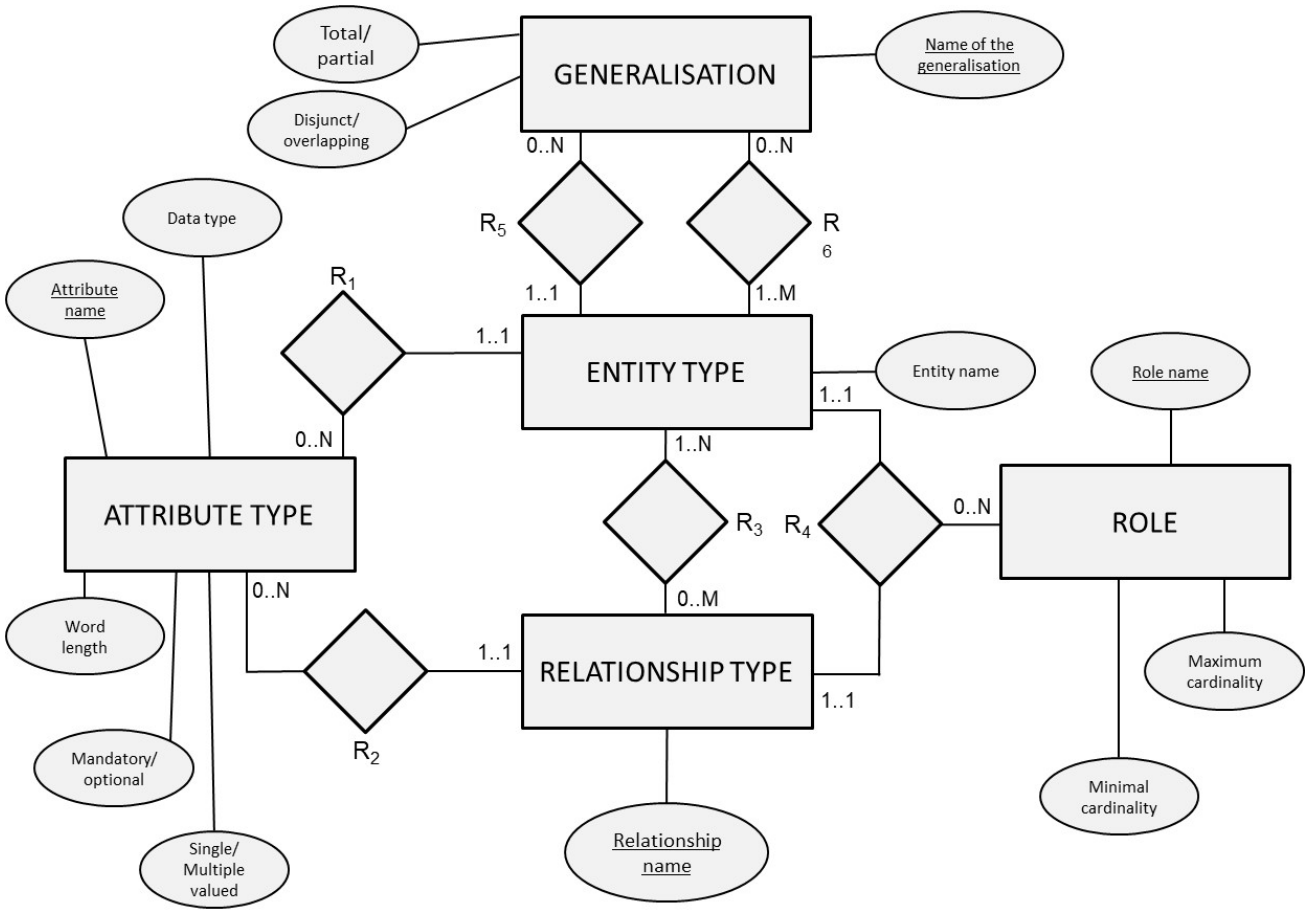
Catalogs and the Role of Metadata

- The catalog provides an important source of information for end users, application developers, as well as the DBMS itself
- Catalog should provide:
  - an extensible metamodel
  - import/export facilities
  - support for maintenance and re-use of metadata
  - monitoring of integrity rules
  - facilities for user access
  - statistics about the data and its usage for the DBA and query optimizer

Metadata Modeling

- A metamodel is data model for metadata
- A database design process can be used to design a database storing metadata
- Design a conceptual model of the metadata: EER model or UML model

Metadata Modeling



Data Quality

- Data quality (DQ) is often defined as ‘fitness for use’
  - » data of acceptable quality in one decision context may be perceived to be of poor quality in another
- Data quality determines the intrinsic value of the data to the business
  - » GIGO: Garbage In, Garbage Out
  - » E.g., obsolete addresses
- Poor DQ impacts organizations in many ways
  - » operational versus strategic level

Data Quality

- DQ is a multi-dimensional concept in which each dimension represents a single aspect or construct, comprising both objective and subjective perspectives
- A DQ framework categorizes the different dimensions of data quality
- Example: Wang et al. (1996)
  - » 4 categories: intrinsic, contextual, representation, access

Data Quality

Category	DQ dimensions	Definitions
Intrinsic	Accuracy	The extent to which data is certified, error-free, correct, flawless and reliable
	Objectivity	The extent to which data is unbiased, unprejudiced, based on facts and impartial
	Reputation	The extent to which data is highly regarded in terms of its sources or content

Data Quality

Category	DQ dimensions	Definitions
Contextual	Completeness	The extent to which data is not missing and covers the needs of the tasks and is of sufficient breadth and depth of the task at hand
	Appropriate-amount	The extent to which the volume of data is appropriate for the task at hand
	Value-added	The extent to which data is beneficial and provides advantages from its use
	Relevance	The extent to which data is applicable and helpful for the task at hand
	Timeliness	The extent to which data is sufficiently up-to-date for the task at hand
	Actionable	The extent to which data is ready for use

Data Quality

Category	DQ dimensions	Definitions
Representation	Interpretable	The extent to which data is in appropriate languages, symbols and the definitions are clear
	Easily-understandable	The extent to which data is easily comprehended
	Consistency	The extent to which data is continuously presented in the same format
	Concisely-represented (CR)	The extent to which data is compactly represented, well-presented, well-organized, and well-formatted
	Alignment	The extent to which data is reconcilable (compatible)

Category	DQ dimensions	Definitions
Access	Accessibility	The extent to which data is available, or easily and swiftly retrievable
	Security	The extent to which access to data is restricted appropriately to maintain its security
	Traceability	The extent to which data is traceable to the source

Data Quality

- Accuracy refers to whether the data values stored for an object are the correct values
  - often correlated with other DQ dimensions
- Completeness can be viewed from at least 3 perspectives:
  - schema completeness: refers to the degree to which entity types and attribute types are missing from the schema
  - column completeness: refers to the degree to which there exist missing values in a column of a table
  - population completeness: refers degree to which the necessary members of a population are present or not

- The consistency dimension can also be viewed from several perspectives:
  - consistency of redundant or duplicated data in one table or in multiple tables
  - consistency between two related data elements
  - consistency of format for the same data element used in different tables

- The accessibility dimension reflects the ease of retrieving the data from the underlying data sources
  - » often involves a trade-off with security

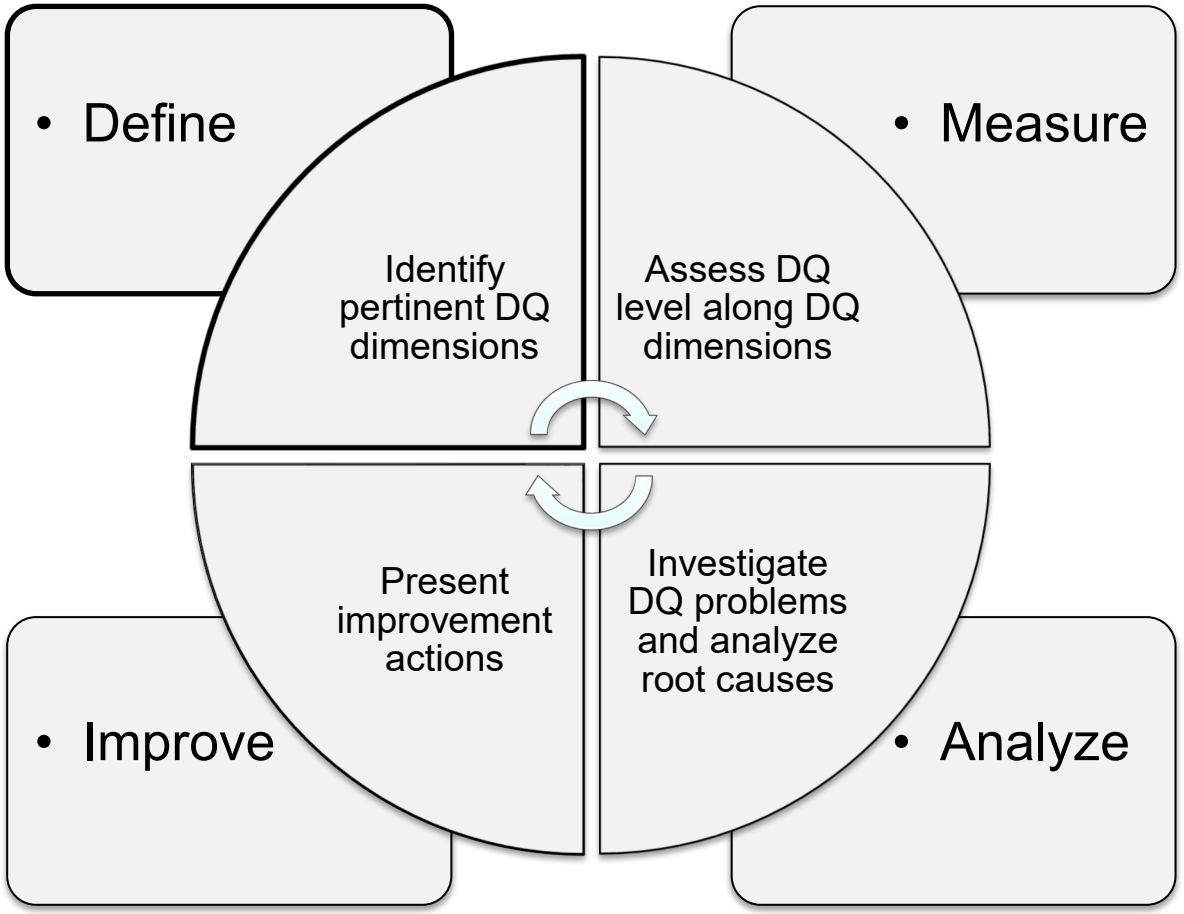
Data Quality

- Common causes of bad data quality are:
  - multiple data sources: multiple sources with the same data may produce duplicates; a problem of consistency.
  - subjective judgment in data production: data production using human judgment can result in biased information; a problem of objectivity.
  - limited computing resources: lack of sufficient computing resources may limit the accessibility of relevant data; a problem of accessibility.
  - volume of data: Large volumes of stored data make it difficult to access needed information in a reasonable time; a problem of accessibility.
  - changing data needs: data requirements change on an ongoing basis; a problem of relevance.
  - different processes updating the same data; a problem of consistency.
- Decoupling of data producers and consumers contributes to data quality problems

Data Governance

- To manage and safeguard data quality, a data governance culture should be put in place assigning clear roles and responsibilities
  - » manage data as an asset rather than a liability
- Different frameworks have been introduced for data quality management and data quality improvement
  - » examples: Total Data Quality Management (TDQM), Total Quality Management (TQM), Capability Maturity Model Integration (CMMI), ISO 9000, Control Objectives for Information and Related Technology (CobiT), Data Management Body of Knowledge (DMBOK), Information Technology Infrastructure Library (ITIL) and Six Sigma

Data Governance



Wang (1998)

Data Governance

- Annotate the data with data quality metadata as a short term solution
  - can be stored in the catalog
  - E.g., credit risk models could incorporate an additional risk factor to account for uncertainty in the data
- Unfortunately, many data governance efforts (if any) are mostly reactive and ad-hoc

Roles in Data Management

- Information Architect
- Database Designer
- Data owner
- Data steward
- Database Administrator
- Data Scientist

Roles in Data Management

- Information Architect (a.k.a. Information Analyst)
  - responsible for designing the conceptual data model
  - bridges the gap between the business processes and the IT environment
  - closely collaborates with the database designer who may assist in choosing the type of conceptual data model (e.g. EER or UML) and the database modeling tool

Roles in Data Management

- Database Designer
  - translates the conceptual data model into a logical and internal data model
  - also assists the application developers in defining the views of the external data model
  - defines company-wide uniform naming conventions when creating the various data models

Roles in Data Management

- Data owner
  - has the authority to ultimately decide on the access to, and usage of, the data
  - could be the original producer of the data, one of its consumers, or a third party
  - should be able to insert or update data
  - can be requested to check or complete the value of a field

Roles in Data Management

- Data steward
  - DQ experts in charge of ensuring the quality of both the actual business data and the metadata
  - perform extensive and regular data quality checks
  - can initiate corrective measures or deeper investigation into root causes of data quality issues
  - can help design preventive measures (e.g. modifications to operational information systems, integrity rules)

Roles in Data Management

- Database Administrator (DBA)
  - responsible for the implementation and monitoring of the database
  - closely collaborates with network and system managers
  - also interacts with database designers
- Data scientist
  - responsible for analyzing data using state-of-the-art analytical techniques to provide new insights into e.g. customer behavior
  - has a multidisciplinary profile combining ICT skills with quantitative modeling, business understanding, communication, and creativity

Summary

- Data Management
- Roles in Data Management

Agenda

1. Session Overview
2. Fundamental Concepts of Database Management
3. Architecture and Classification of DBMSs
4. Organizational Aspects of Data Management
5. Summary and Conclusion

Agenda

- Fundamental Concepts of Database Management
  - Applications of Database Technology
  - Key definitions
  - File versus Database Approach to Data Management
  - Elements of a Database System
  - Advantages of Database Systems and Database Management
- Architecture and Classification of DBMSs
  - Architecture of a DBMS
  - Categorization of DBMSs
- Organizational Aspects of Data Management
  - Data Management
  - Roles in Data Management

Any Questions?