Lecture 1. Database Concepts and Architectures (Chapter 1 and 2)

Alisa Lincke

alisa.lincke@lnu.se

What database is and what it does

Database is a collection of related data.

Data are known facts that can be recorded and have an implicit meaning. (e.g., people names, mobile phones, addresses)

What it does:

- Store data
- Provide an organizational structure for data
- Provide a mechanism to interact with data (e.g., querying, creating, modifying, and deleting data, or CRUD (create, read, update, delete))
- Can store data and relationships that are more complicated than a single table can

A Database Properties

A database represents some aspect of the real world (e.g., University, Shop, Hospital)

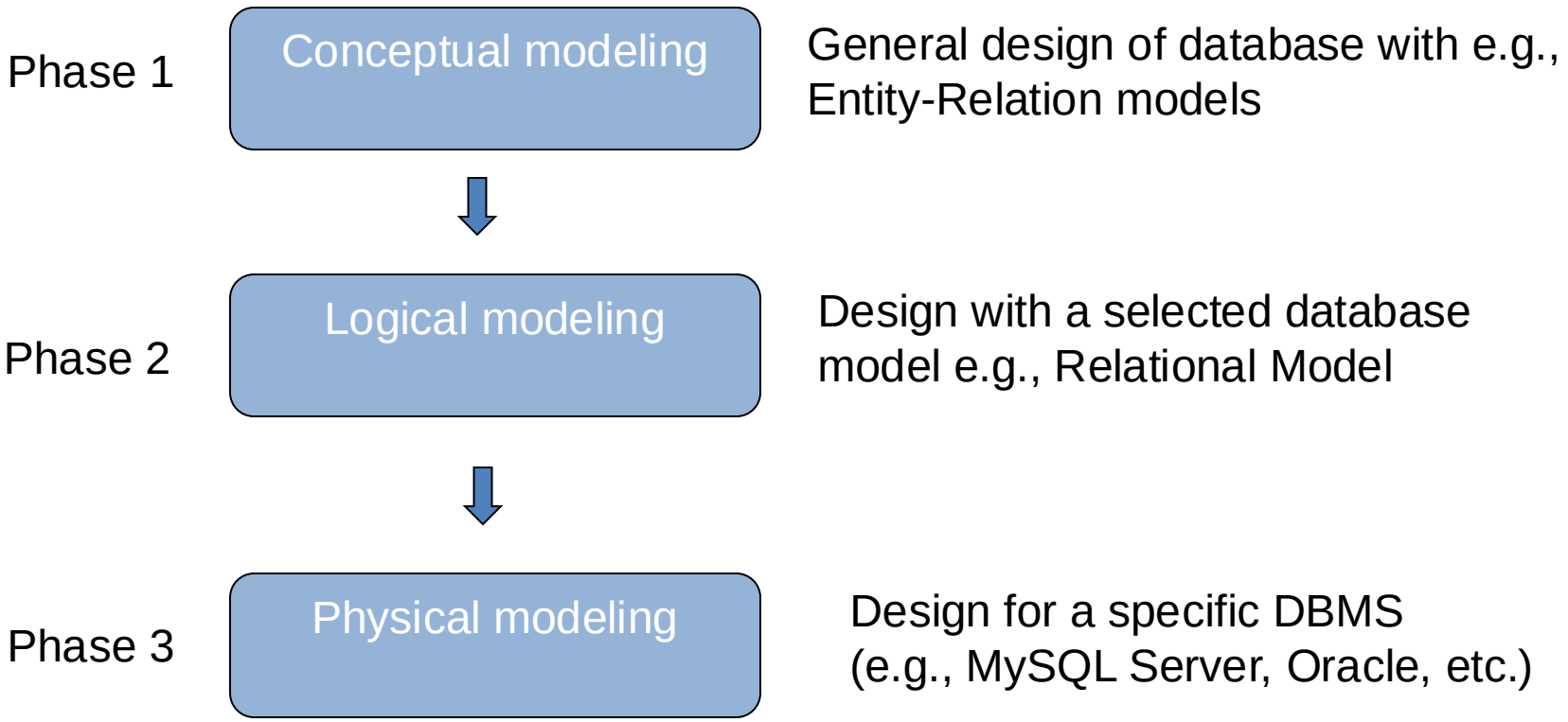
A database is a logically coherent collection of data with some inherent meaning (**relationships** between data)

A database is **designed**, **build**, and **populated** with data for a specific purpose and for target **users**, and **applications**

A database can be of any **size** and **complexity**

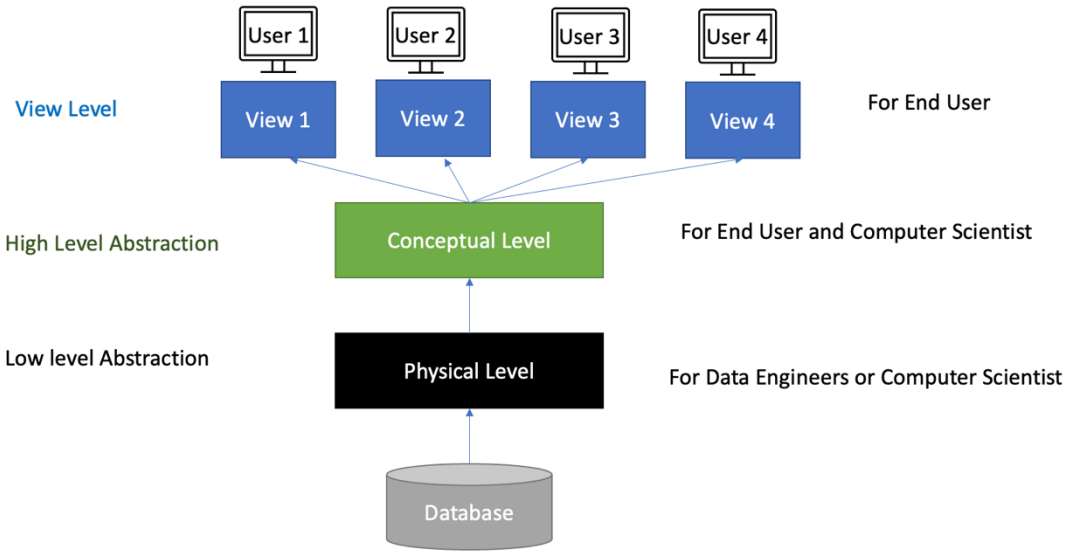
Traditional database applications	Non-traditional database applications
• Store textual or numeric related data • Supports relationships in data • Examples: MySQL (relational database) , Oracle, PostgreSql	• Store non-traditional data (images, tweets, files, videos, etc.) • Not necessarily support relationships in data • Examples: NoSQL (non-relationships data), Graph databases, Cloud storage (Big Data technologies for storing and managing big data)

Database Modelling in Three Phases



Data Abstraction

Data abstraction refers to suppression of details of data organization and storage (e.g., different users can perceive data at their preferred level of detail) and aims to support **data independence**.



Data Independence

Data independence is a abality to make a change in one level of abstraction without having to change the next higher level.

Two types of data independence:

- **Logical data independence** (conceptual level) is ability to make changes in conceptual level wihtout making changes to view level (application programs). For example, we can expand the database (add new data), or change constrains, or reduce the data in database.
- **Physical data independence** (physical level) is ability to change physical logic without having to change the conceptual level and view level. For example, migrate database to another drive disk, or computer to improve the performance.

Database Users

Database Administrators: are administrating the content, the DBMS and realted software. The responisibilities are providing the authoriable access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed, security breaches, and system response time.

Database Designers: are responsible for identifying the data to be stored in the database according to user requirements (data abstraction, data modeling, indetifying entities and their relationships).

End Users: are the people whose jobs requires access to the database for querying, updating, and generating reports.

Software Engineers: are developers which implements requirements defined by database designers or on the database design phase. Then they, test, debug, and maintain these implemneted queries/interactions to database.

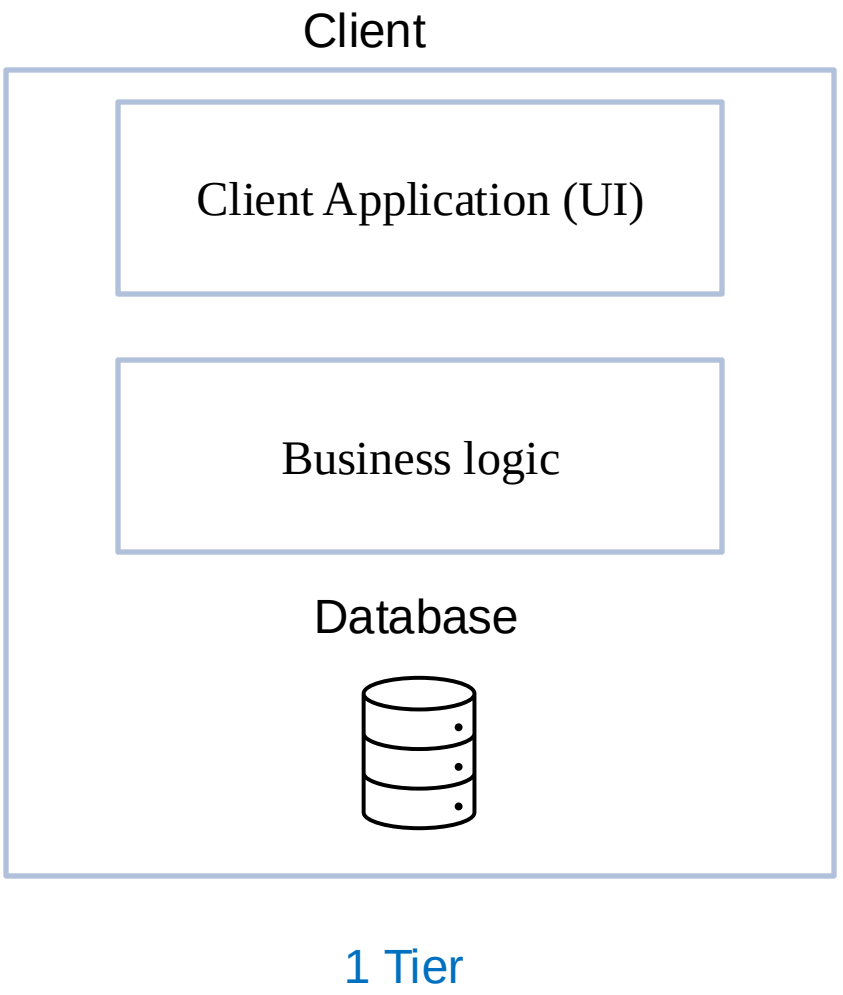
Database Components

1. **Database management System** (DBMS) is software to create and maintain a database.
2. **Database Server** refers to the hardware or combination of hardware and software that managed by the DBMS.
3. **Application** is a software for end-users with some user-interface to interact with the data.

DBMS Architecture: 1-Tier

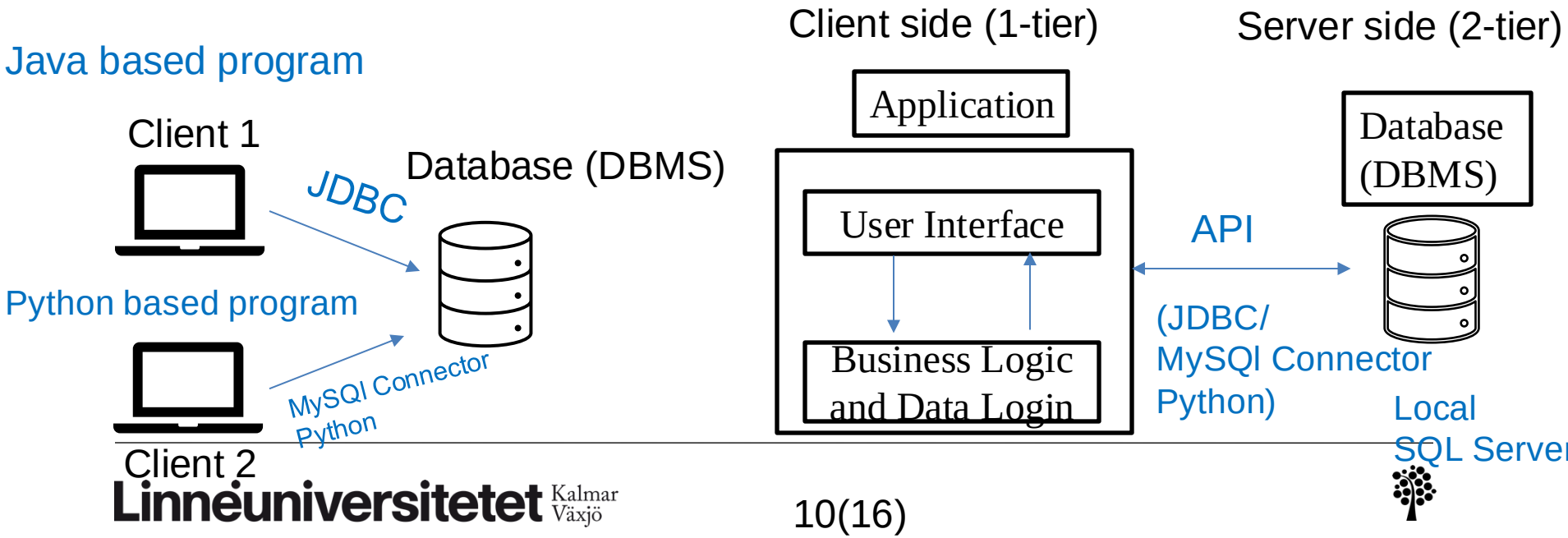
1-tier architecture is where *client* (is UI software provided by the DBMS), *server* (is a client's computer/machine), database (data itself) stored on one computer/machine

Examples: Microsoft Access, native mobile application (which does not use Internet)



DBMS Architecture: 2-Tier

2-tier architecture Client/Server architecture is centralized architecture consist of client DBMS software and server database software. The application programming interface (APIs) allows to manipulate and communicate with database only if both client and server installed. The application program (user interface and business logic) implemented on client side, the server side has connection to database and processing queries to database.

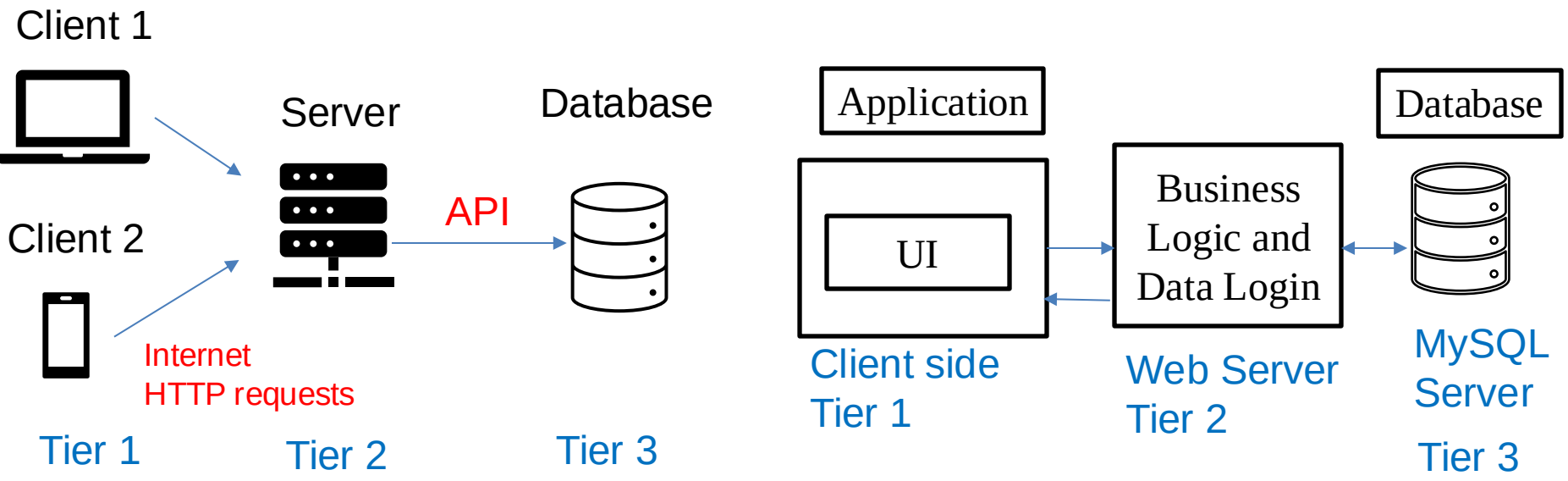


DBMS Architecture: 3-Tier

Server is a centralized computer that provides services to all connected clients through network connection. For example, web server, file server, etc. each has a specific work to provide services to each client. Has installed server software.

Client is laptop, tablet, mobile phone provided with the appropriate user interfaces to use these services, and provide local processing power to run local applications. Has installed client software.

Example, any web site on the Internet



Other Database Classifications

Centralized	Decentralized	Distributed
Has a central control authority (only one). The data and data processing implemented one one server/computer. Examples: MySQL, Oracle, PostgreSQL, etc.)	Has multiple controlling authorities (multiple) Example: Blockchain-based database (BigchainDB, Ethereum, etc.)	The data and data processing (workload) distributed over many computers/servers and has single owner/authority. Example: Apache Ignite, Apache Cassandra, Apache HBase, Couchbase Server, Amazon SimpleDB
Library Management Systems, Content management systems, etc.	Cryptocurrencies, Supply Chain Management, Voting Systems	Social Media Applications, Streaming data applications (IoT), Telecommunications, Google Docs, others.

Database Types

- Hierarchical Database
- Network Database
- Relational Database
- Object-Oriented Database
- NoSQL databases
  - Document-Oriented Database
  - Columnar Database
  - In-Memory Database
- Graph Database
- And more

Selecting a database

1. Analyze the data (structure, volume, complexity, frequency, identify data relationships)
2. Analyze application requirements:
   - Number of users, and amount of data for read/write operations per user
   - Scalability Requirements (if data volume and traffic growth fast use horizontal scaling)
   - Query and Performance Requirements (analyze types of queries will frequently execute)
   - Assess the importance of data consistency and transaction integrity.
   - Flexibility and Schema Requirements (how often the schema will evolve)
   - Security
   - Cost, Maintenance

Key Terms for Exam

data abstraction: is process to provide different views (details) on the data structure

data independence: the flexibility of the DBMS to make changes on one level of abstraction without making changes on the next level of abstraction (or data representation/view level)

three phases of the database design: conceptual, logical, and physical.

conceptual level (high level abstraction): is *describes* database requirements in text and diagrams using a set of concepts (e.g., entities, attributes, tables, keys, schema) in order to communicate with the customers (not technical people).

physical level (low level abstraction): is *describes* the physical storage structure of the database (defines internal database structure (relational data model), file organization, indexing techniques, etc.)

difference between conceptual and physical level: conceptual level describes database with text and ER diagrams without knowing which data model will be used (e.g., relational data model, not relational data model) while physical level describes in detail the data structure using a specific database type.

conceptual data models: a collection of concepts (e.g., entities, attributes) to describe the database for communication with customers (not technical people) to get early feedback if we did not miss anything and understood the requirements correctly.

physical level data models: relational data model (internal schema), file structure, DBMSs architecture (1-tier, 2-tier, 3-tier)

Key Terms for Exam

1-tier architecture: the database, user interface and application logic are all in a single system (e.g., native mobile application, SQLite). The user interacts with the database directly, there is no separate client or server.

2-tier architecture: *client* (desktop application such as Java-based, Python-based) and *server* (MySQL Server). The user interface and application logic are on the client side (as Java-based, Python-based application). Client communicates with database server using APIs.

3-tier architecture: *client* (front-end web app), *web server* (application logic, back-end web app), and *database server* (data stored on a web server). Clients does not have direct access to the database, clients communicates with server over HTTP protocol, and web server application communicates with database using APIs.

difference between ***centralized*** and ***decentralized*** databases is that centralized approach has one central server where the database is installed and one database, while in the distributed database approach, there are many servers installed and same database is replicated in all nodes.

difference between ***2-tier*** and ***3-tier architectures*** is that 3-tier architecture has extra separation layer (web server application (business logic) located at server side, client side has only user interface application) and two connections (client-web server connection over https, and web server-database server connection via APIs).