# PRINCIPLES OF DATA ORGANISATION

Introduction

# DATA ARE HERE TO STAY …

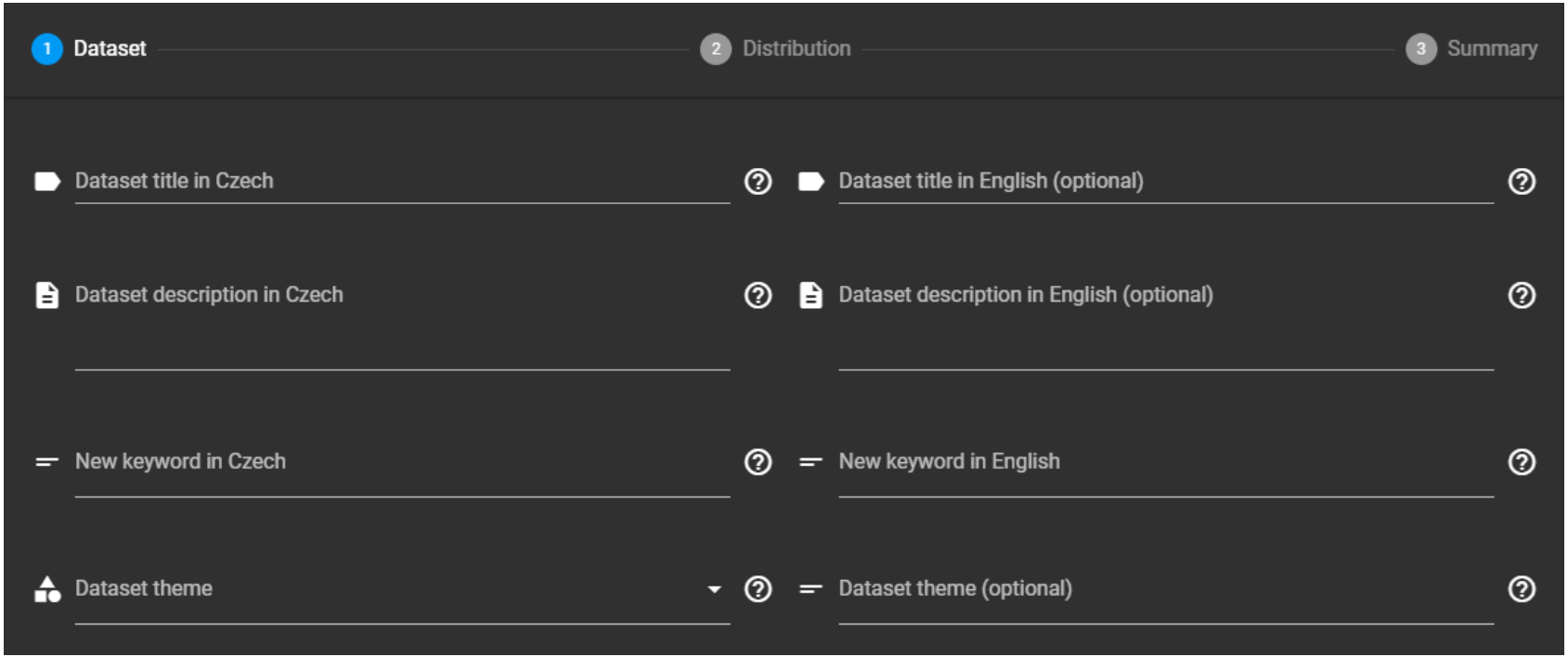
“Software comes and goes, data is forever.”

Ondřej Felix, Open data in public administration 2019

# HOW DO WE USE DATA?

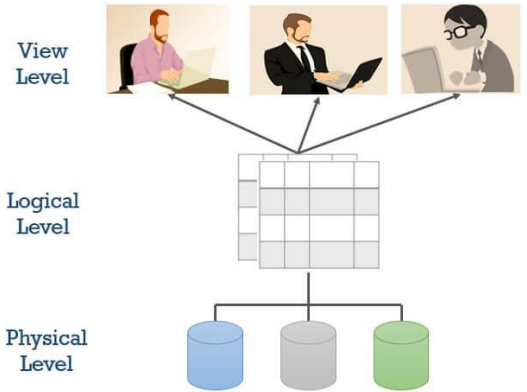
Technology vector created by starline – www.freepik.com

# HOW DO WE USE DATA?



# VIEW OF DATA IN DBMS

- Data abstraction - allows developers to keep complex data structures away from the users
- Several levels:
  - Physical level (internal level)
    - how the data is stored in the hardware
      - Primary files, indices, …
  - Logical level
    - Tables, graphs, documents, …
  - Conceptual level
    - Entities, properties, relationships, …
  - View level (external level)
    - Data in which the user is interested
    - Many views of the same data



# WHERE IS YOUR DATA STORED INTERNALLY?

Small data => main memory
Big data => secondary device

When dealing with large amount of data, classical in-memory data structures fail because they consider flat memory hierarchy

Need for specialized data techniques and structures
for data not present in the main memory

memory vs. storage → nanoseconds vs. milliseconds

# DATA LIVE IN BLOCKS


Each of the drives attached to this computer has a block size of 4096; each time we read one byte, the drive gives us a block containing 4096 bytes.

If we organise our data structure carefully, each disk access could yield 4096 bytes that are helpful in completing whatever operation we are doing.

# MEMORY MODELS

# I. RANDOM ACCESS MEMORY MODEL

Traditional model where the run time is based on the number of instructions concerning the size of the input n.

Assumptions:

- Large enough main memory to store all the data
- Memory allocation in unit time

# RANDOM ACCESS MEMORY MODEL – APPLICATIONS

- In-memory database - primarily relies in main memory
  - Faster, more predictable performance (no seek)
  - Expensive, volatility of devices

- SAP HANA

  Requires tailored application for in-memory databases

- ORACLE DATABASE

  Allows to switch to in-memory mode without the need of migration

# II. EXTERNAL MEMORY MODEL

- Also known as I/O model or disk access model
- Counting the number of memory transfers between cache (infinitely fast) and disk (external memory)
  - The instructions are much faster
- Data between disk and cache can be transferred only by blocks

- **Our main focus during the course**

# III. CACHE-OBLIVIOUS MODEL

- Full memory hierarchy
  - Each memory can have different characteristics
  - Registers, caches, main memory, …, hard drives, …
- Cache-oblivious algorithm - does not know the size of its cache or the block size
  - No program variables dependent on hardware configuration parameters
  - Designed to take advantage of a processor cache without having the size of the cache
- Typically, recursive divide-and-conquer algorithm
  - Divided into smaller and smaller subproblems
  - Eventually, one reaches a subproblem size that fits into the cache,
    - regardless of the cache size