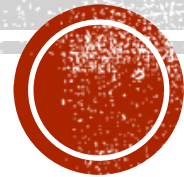


PRINCIPLES OF DATA ORGANISATION

File Organization Theory



MOTIVATION

- ⌘ How can we organize data records?
- ⌘ Some theory first ...



TERMINOLOGY

Data record

- ↳ Representation of an application object
 - ↳ Person, car, product, ...
- ↳ Set of fields
 - ↳ Elementary unit of data
 - ↳ Age, length, ...
 - ↳ Fixed/variable length
- ↳ Attribute is a field with a domain (data type)
- ↳ Key
 - ↳ Identifier of a record

File

- ↳ Named collection of records

Database

- ↳ Collection of related data (named files) in secondary memory



DATA RECORD

Logical level

- ↳ Attribute set

Physical level

- ↳ A physical representation of a logical record of size **R** (bytes) in a medium
- ↳ Contains additional metadata
 - ↳ E.g., record delimiters, internal structure information
- ↳ Records are stored in blocks of size **B** (bytes)



DATA RECORD

Fixed length

- 🔗 File header contains:
 - 🔗 number of records
 - 🔗 length of each field
- 🔗 Record can be accessed using the record number (index)

Variable length

- 🔗 Variable length of the attributes
 1. Names of different sizes
 2. Similar objects related to a nonuniform set of data
 - 🔗 E.g., different attributes for different types of employees
 3. Optional attributes
 - 🔗 E.g., product picture
 4. Attributes holding records
 - 🔗 E.g., order with multiple items, employee with multiple phone numbers
- 🔗 Workaround is to set maximum length for a given field



RECORD BLOCKING

Blocking factor

- Number of records in a block
- $b = \lfloor B/R \rfloor$
 - B = block size
 - R = record size
- Remember: We read blocks, not records

Basic division based on blocking

- non-blocked records
 - 1 record fits 1 block
 - easy manipulation
- blocked records
 - N records fit 1 block
 - Most efficient
- overflow records
 - 1 record fits N blocks
- Blocked/overflown
 - records written without respecting the blocks boundaries
 - suitable for variable-length records or texts



RECORD BLOCKING STRATEGIES

Fixed blocking

- ❧ Fixed-length records
- ❧ Put maximum records in a block
- ❧ Possible internal fragmentation
 - ❧ Unused remaining space

Variable-length spanned blocking

- ❧ Variable-length records
- ❧ A record can span multiple blocks
- ❧ Continuation is indicated by a pointer to the next block
- ❧ Hard to implement
- ❧ Needs more time to read records in 2 or more blocks

Variable-length unspanned blocking

- ❧ Variable-length records
- ❧ No spanning
- ❧ Each record occupies a block (starts at the beginning of a block)
- ❧ Unused space is wasted
 - ❧ Possible high internal fragmentation



FILES

- ↳ Collection of record stored in the secondary memory
 - ↳ Reading = block transport to the main memory
 - ↳ Modification = read → edit in memory → save
- ↳ Records are identified using file keys
 - ✦ **File key** $K = \text{set of attributes } \langle A_{j1}, \dots, A_{jk} \rangle$
 - Its values $\langle a_{j1}, \dots, a_{jk} \rangle$ uniquely identify a record
 - ↳ **Record key**
 - ✦ One of the keys is denoted as a primary key
 - Should be artificial

Homogeneous

- ↳ Store fixed size records of the same type

Non-Homogenous

- ↳ Either with variable size or with different type



FILES — OPERATIONS

Modification

- ✂ Insert
- ✂ Update
- ✂ Delete

Querying

- ✂ Find
 - ✂ Find a record within the file
- ✂ Fetch
 - ✂ Loads a record into the main memory

Formation/Termination

- ✂ Create/Remove

Maintenance

- ✂ Reorganize/Rebuild
 - ✂ Not all changes are immediately projected into the underlying file organization
 - ✂ Optimization



QUERYING FILES

One-dimensional queries

- cars with **age** > 35
- cars with **color** = 'red'

Multi-dimensional queries

- Total match
 - All attributes specified
 - age** = 12 & **color** = 'red'
- Partial match
 - age** = 12
- Total interval match
 - 12 < **age** < 25 & **color** in ('red', 'blue')
- Partial interval match
 - 12 < **age** < 25



FILE ORGANIZATION

- ⌘ How to organise a set of records in a file and how to access them
- ⌘ The description of the **logical memory structure** together with **algorithms** for handling that structure
 - ⌘ Can involve multiple files
- ⌘ Optimal choice of an organization depends on the usage (operations, amount of data, ...)



FILE ORGANIZATION — LEVELS

Logical schema

- ↳ Algorithms
 - ↳ Secure and optimal manipulation with blocks/files
- ↳ Logical blocks (pages) in memory
 - ↳ Structure
 - ↳ Relations
 - ↳ Content
 - ↳ Manipulation
- ↳ Logical files
 - ↳ How the logical pages are related to each other
 - ↳ Primary file
 - ↳ Data records
 - ↳ Auxiliary files
 - ↳ Indices, metadata

Physical schema

- ↳ Mapping between logical schema and physical pages
- ↳ One logical file can span multiple physical files and vice versa
- ↳ E.g., an area of a magnetic disc

Implementation schema

- ↳ Implementation of the physical files shielded from the logical level by OS
- ↳ E.g., particular track, sectors, etc.

