Lancaster University

SCC.201
Database Management Systems

2023 - Week 6 – Physical Storage And Access - Files.

Uraz C Turker & Ricki Boswell-Challand

Further Reading: Elmasri ch.16

Storage of databases

Lancaster University

- Most DB data resides on secondary storage
  - HDD/SSD
  - May be too large to reside entirely in main memory (e.g. H-Unique core datasets > 250GB and will grow to several TB)
  - Secondary storage costs orders of magnitude lower than main memory, but slower to access
  - Often not all database data required frequently – access from disk as required
  - Secondary storage offers persistence
  - Trade off between storage capacity, robustness and speed of access
    - Physical access characteristics affect access latency and transfer rate of data

Storage of databases

Lancaster University

- Main memory DBs do exist
  - Entire database held in main memory (along with OS, DBMS and possibly other applications)
  - Suited for real-time applications requiring extremely fast response times
    - E.g. Telephone network routing, high-frequency trading
  - Extremely expensive for large datasets
  - Many (most?) applications do not require this level of response
  - Beyond scope of this module

Storage of databases

Lancaster University

- Techniques for storage of large amounts of data are important to understand
  - Database designers and administrators need to consider advantages and disadvantages of each technique
  - Physical database design involves selecting appropriate organisation techniques to select for specific application requirements – domain knowledge as well as technical knowledge needed!
  - DBMS designers need to provide efficient implementations of physical data organisation techniques for database designers and admins to utilize.

Files and records

Lancaster University

- Data is organised on disk into files of records
  - Record is a collection of related data items
    - e.g. Personnel record may contain forename, surname, DOB, NI number etc
    - Each item consists of one or more bytes of data
    - Each data item corresponds to a particular field of the record

Forename	Surname	DOB	NI_No	Salary
Susan	Smith	27/04/1989	ND783674V	48,500
Jo	Kerr	16/11/1995	BH478256T	39,000

Files and records

Lancaster University

- A record type is a collection of field names and their corresponding data-types
- Data types of fields are standard data types such as integer, float, character strings, date etc
- Number of bytes required to store data items of each particular type if fixed for a given computer system

Forename	Surname	DOB	NI_No	Salary
String	String	Date	Char[9]	Float

Files and records

Lancaster University

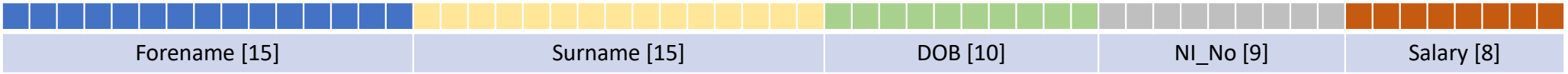
- A file is a sequence of n records
  - Often all records in a file are of the same record type but this does not have to be the case

- Fixed length records – every record in file is exactly same size (in bytes)
- Variable length records – file contains records of differing lengths

Fixed length records

Lancaster University

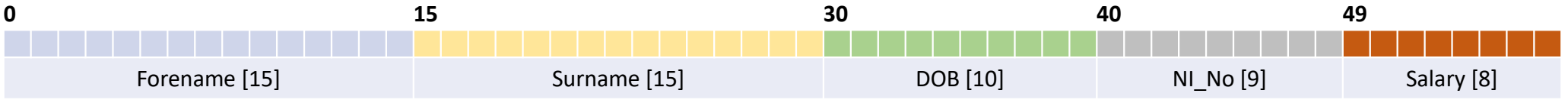
- Consider our personnel record
- If forename and surname have a maximum defined length of 15 characters each, a record can be held in a known, fixed number of bytes:



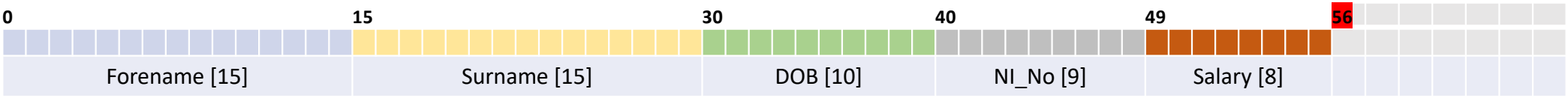
Fixed length records

Lancaster University

- Starting byte position of each field can be identified relative to start of record



- Similarly, start position of next record can be identified relative to position of



Variable length records

Lancaster University

- Different records in a file may have different sizes in bytes for a number of reasons
  - Records of same record type but have one or more varying length fields (e.g. name fields of employee record)
  - Records of same type but a field may have multiple values for a record (e.g. multiple contact phone numbers)
  - Records of same type but one or more fields may be optional

Variable length records

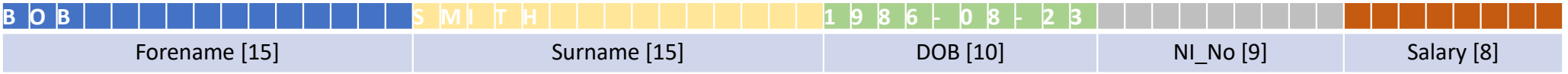
Lancaster University

- Records of different types clustered together for performance and retrieval purposes
  - E.g. Student and grade records might be held together
  - Grades of each student follow directly after personal information

Variable length records

Lancaster University

- A file containing variable length records can be represented as a fixed-length records file
- Values that do not fill maximum available field length can be padded
- Optional fields can be included with empty (NULL) values if no value exists
- Repeating fields can be allocated as many ‘spaces’ as maximum number of values a record can take
- All of these methods waste space on disk!



Variable length records

- Fields in variable length records can be terminated by special separator character

Variable length records

Lancaster University

- Fields in variable length records can be terminated by special separator character

- Other methods for encoding variable length fields may include
  - Fields as name/value pairs <field name, field value>
  - Field length (in bytes) can be stored preceding field value

Record blocking & disk storage

Lancaster University

- Records of a file must be allocated to disk blocks
  - Block is data transfer unit between disk and main memory
- Three possibilities for block sizing
  - Block size > record size
    - Block may contain several records
  - Block size < record size
    - Record is stored across multiple blocks
  - Block size = record size
    - Exactly one record per block

Record blocking & disk storage

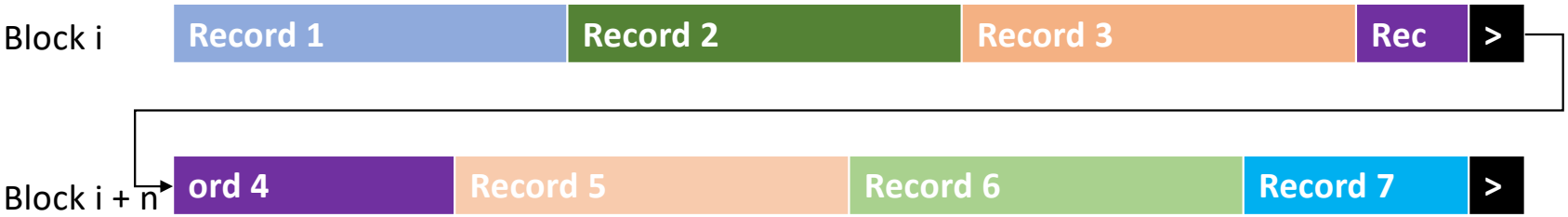
Lancaster University

- Suppose block size is B bytes and a file contains fixed length records of size R bytes
- If B > R we can fit bfr = $\lfloor B / R \rfloor$ records per block
- Floor function = $\lfloor (x) \rfloor$ rounds the number x down to an integer
  - Thus $\lfloor 12.7 \rfloor = 12$
- bfr is known as the blocking factor
- R may not divide B exactly.
  - There will be unused space in each block equal to B- (bfr * R) bytes

Record blocking: Spanned records

Lancaster University

- If a block does have enough remaining space to store complete record, situation may be handled in one of two ways:
  - Spanned records may be spread across two blocks, first block has pointer to block containing rest of record



Record blocking: Unspanned records

Lancaster University

- In this case records are not allowed to cross block boundaries
-

Record 1	Record 2	Record 3	Empty
Record 4	Record 5	Record 6	Empty
Record 7	Empty		

Allocating File Blocks On Disk

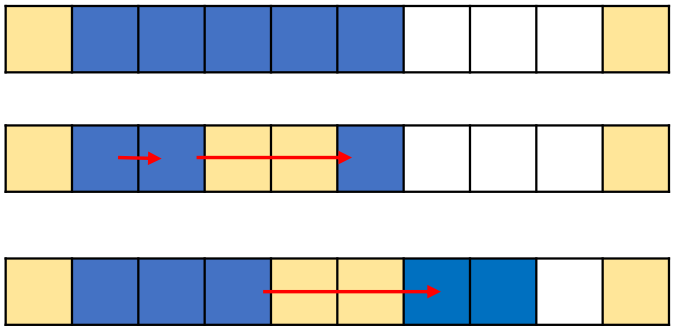
Lancaster University

- Database files typically have an initial allocation of blocks on disk
- As data growth occurs (and it nearly always does!) files need be able to grow to accommodate the enlarged data.
- Block allocation routines need to balance performance, flexibility and space efficiency
- Disk space is shared with any OS and other application files that are using the same disk

Allocating File Blocks On Disk

Lancaster University

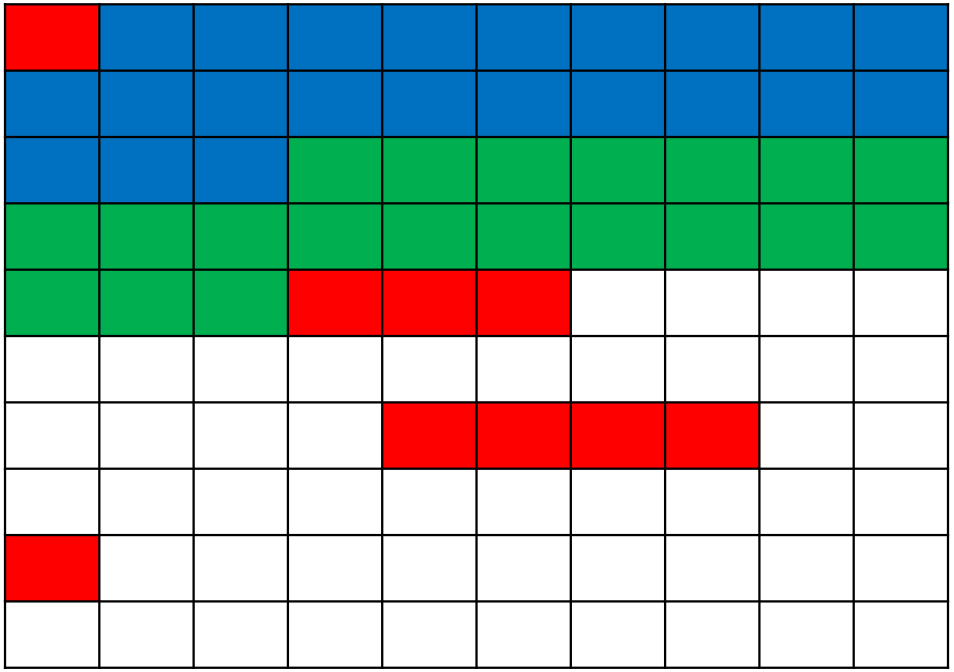
- Various ways of allocating file blocks to disk blocks
  - Continuous allocation
    - File blocks allocated to consecutive disk blocks
  - Linked allocation
    - Each file block contains a pointer to next file block
  - Cluster allocation
    - Combines continuous and linked allocations
    - Allocate clusters of blocks and link them with pointers
  - Indexed Allocation
    - One or more index blocks contain pointers to actual file blocks



Allocating File Blocks On Disk

Lancaster University

- Continuous allocation



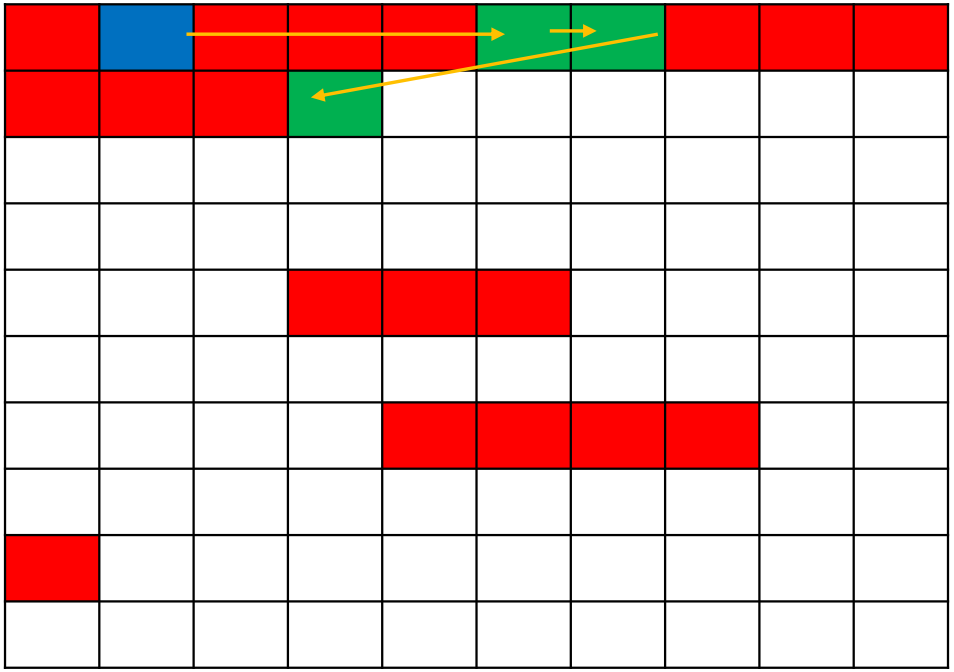
Empty disk block
DB file block
Used disk block
File growth space

- +ve: Very fast reading of whole file as blocks are contiguous
- -ve: File expansion difficult due to used blocks on disk

Allocating File Blocks On Disk

Lancaster University

- Linked allocation



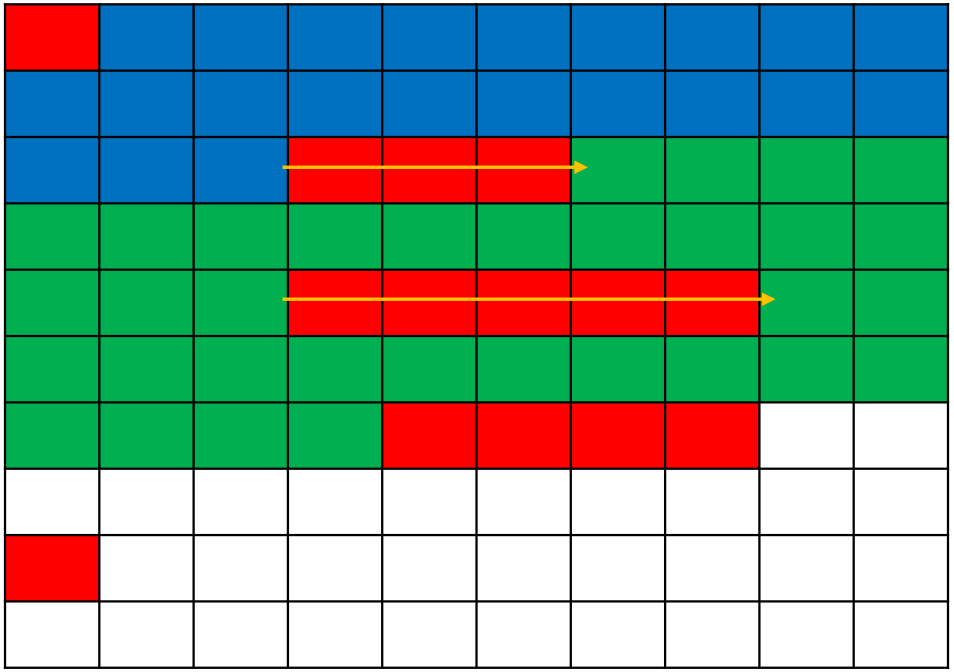
Empty disk block
DB file block
Used disk block
File growth space

- +ve: Very easy to expand file, just allocate next free block and add a pointer
- -ve: File reads are slower, especially with magnetic disks

Allocating File Blocks On Disk

Lancaster University

- Continuous allocation



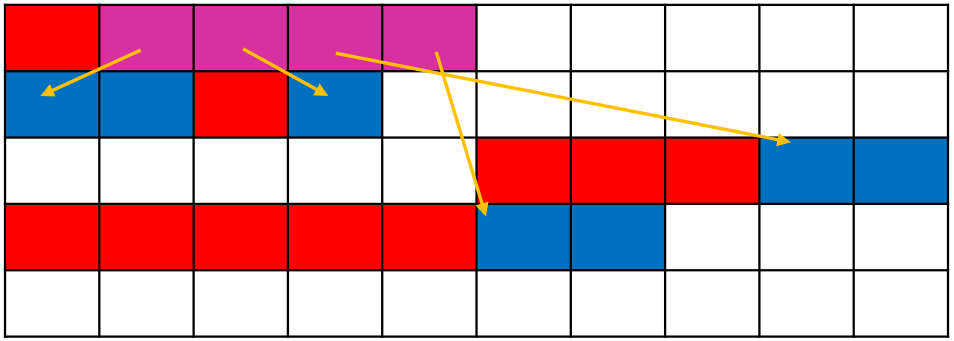
Empty disk block
DB file block
Used disk block
File growth space

- Achieves a balance between ease of expansion and read performance

Allocating File Blocks On Disk

Lancaster University

- Indexed allocation



Empty disk block
DB file block
Used disk block
Index Block

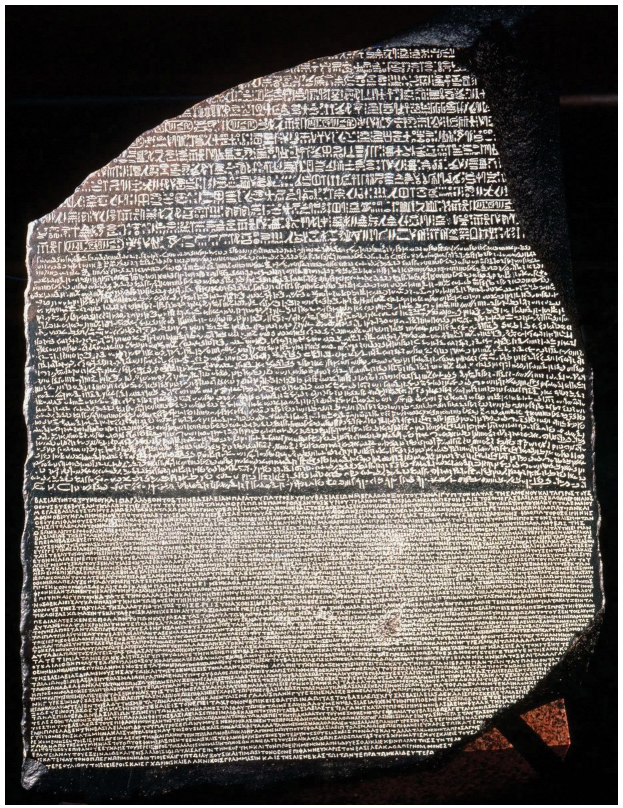
- Read index blocks to find pointer to blocks containing desired data
- Analagous to using a library card index
- Indexing covered later in course

File Headers

Lancaster University

- Also known as file descriptor
- Contains information needed by programs accessing records in the file
  - Information to determine disk addresses of file blocks
  - Record format descriptions
    - For both fixed and variable length records
    - Order of fields
    - Type codes
    - Field & record separators
- The Rosetta Stone for reading a file

Going off on a tangent…



- Rosetta Stone
- Egyptian ‘Stele’ or stone slab with engraved texts
- Discovered in 1799
- Contains inscriptions which provided the key to deciphering Egyptian hieroglyphics

Typical File Operations

Lancaster University

- Operations on a file grouped into two groups
  - Retrieval Operations
    - Do not change data in file
      - Locate records so field values can be read and processed
  - Update Operations
    - Modify the data file
      - Insertion or deletion of records
      - Alteration of field values within a record or records
  - These are the underlying file operations for which SQL SELECT/INSERT/UPDATE/DELETE statements provide an abstraction

Typical File Operations

Lancaster University

- Open
  - Prepares file for reading/writing
  - Allocates buffers (usually minimum of 2) to hold file blocks from disk
  - Retrieves file header
  - Set file pointer to beginning of file
- Reset
  - Set file pointer back to beginning of file

Typical File Operations

Lancaster University

- Find/Locate
  - Searches for first record satisfying search condition
  - Transfers block containing matched record into main memory buffer (if not already in memory)
  - Points file pointer to fetched record in file
- FindNext
  - Same as find, but gets the *NEXT* matching record in file

Typical File Operations

Lancaster University

- Read/Get
  - Copies current record from buffer to program variable
  - May also advance file pointer to next record in file
    - May cause next block to be read from disk
- Delete
  - Deletes current record
  - Updates file on disk to reflect deletion
- Modify
  - Modifies some fields for current record
  - Updates file on disk to reflect changes

Typical File Operations

Lancaster University

- Insert
  - Locates block where record to be inserted
  - Transfers block to main memory buffer
  - Writes record into buffer
  - Writes buffer to disk
- Close
  - Releases all buffers
  - Performs any other cleanup operations required
- All of the previous operations (except Open and Close) are record-at-a-time operations

Typical File Operations

Lancaster University

- Set-at-a-time operations
  - FindAll
    - Locate all records that satisfy a search condition
  - FindOrdered
    - Retrieves all records in a file in a specified order
  - Reorganise
    - E.g. Reorder file records based on a specified field value

File organisation

Lancaster University

- Primary file organisation
  - How records are physically placed on disk
  - Determines how records are accessed
- Secondary file organisation
  - Efficient access to records
  - E.g. index structures (covered later in course)

Primary organisation: unordered records

Lancaster University

- Also known as heap files
- Simplest most basic type of organisation
- Records placed in file in insertion order
  - New records placed at end of file
- Inserting new record very efficient
  - Address of last block kept in file header
  - Last file block copied to buffer
  - New record added and written to disk

Primary organisation: unordered records

Lancaster University

- Searching for record very expensive
  - Linear search required
- Deletion of record expensive and inefficient
  - Linear search to find block with record to be deleted
  - Copy block to buffer
  - Delete record from buffer
  - Rewrite block to disk
  - Leaves unused space on disk
  - Large no. of deletes leads to much wasted space

Primary organisation: unordered records

Lancaster University

- Alternative deletion approach
  - Use deletion marker
  - Extra bit set to mark records as deleted
  - Search operations ignore deleted records
- Unordered records require periodic file reorganisations to reclaim unused space

Primary organisation: ordered records

Lancaster University

- Physically order records in a file based on one of their fields (called ordering field)
  - Ordering field called ordering key if it is a key field of file
    - i.e. unique value for each record

Primary organisation: ordered records

Lancaster University

Block 1

Surname	Forename	DOB	NI_No	Salary
Abbot	Alan			
Abbot	Julie			
Barker	Simone			

Block n

Surname	Forename	DOB	NI_No	Salary
Smith	Karen			
Turker	Uraz			
Wilson	Charlotte			

Primary organisation: ordered records

Lancaster University

- +ve:
  - Reading records in order defined by key extremely efficient
    - No sorting required
  - Finding next record in order requires no additional block access unless current record is last in block
  - Using search conditions based on ordering key results in faster access

Primary organisation: ordered records

Lancaster University

- -ve:
  - No advantage for access based on values of non-ordering fields
  - Insertion and deletion of records very expensive
    - Records must remain physically ordered
  - E.g insert:
    - Find correct position in file
    - Make space and insert record
    - Very expensive operation for large files
  - Modification of a record
    - Record may change position in file > equivalent to deletion + insertion

Conclusion

Lancaster University

- Physical file organisation is a complex process
- The DBMS abstracts a large amount of this away from the DB admin/designer allowing them to focus on implementing applications
- Understanding of underlying storage structures allows us to optimise our designs