Persistent Storage

- Datastructures and Algorithms
L 02: Direct Access to Data
Questions (1)

- Recording of informations?
- Use of recorded informations?
  - Machines which are able to read recorded I.?
- Persistent data today, where?
- Why do we need conventions / formats?
- What is record oriented data processing?
  - Give an example of a record structure
  - How are punch cards sorted?
Questions (2)

- What is a dataset?
  - Size is described in which units?
  - What is an extent?
  - How are datasets allocated?
  - What is the reorganization of a dataset?

- How are datasets structured?

- What is a PDS?
Questions (3)

- What is a FCB?
  - Operations on a FCB?
  - Where is the buffer, who allocates it?
  - Where is the Cache?
  - What is a Linkname?

- What is a Catalog?

- How is the consistency of data guaranteed?
Records in a Dataset

• What is possible with the records of a dataset?
  – Read and write sequentially
  – Read/write of a certain record specified by number
    • Only with F* format, a record number is required
  – How to read a certain record in V* format?
    • Sequential read from the beginning, counting records ...
  – How to write an arbitrary record in a V* dataset?
    • Sequentially finding the record (by counting), replacing it with the new record, perhaps copying the rest of the dataset ...
Random Access a F* format dataset

• Task: read a certain record
  – Specified by the number of the record

• Implementation:
  – Compute the block number (from record number)
  – Read the block
    • (if not buffered already)
  – Cut out the record from its position
Random Access (2)

- Task: Write a record
  - given the new content and the record number

- Write the record:
  - Compute block number
  - Read block (when using FB or VB format)
  - Update the block with new record contents
  - Write the block back to disk

*random access also called direct access file (DA)*
Applications and direct access files

- Is a direct access file enough for everything?
  - Search for a employee number
  - Search for a certain name
  - Read records with a certain field combination
    - F.e. which employees are male and have kids?

Use the record number like a pointer?
Applications and direct access files

• Is a direct access file enough for everything?
  - Search for a employee number
  - Search for a certain name
  - Read records with a certain field combination
    • F.e. which employees are male and have kids?

=> No!
How to use arbitrary keys?

- Implementation hidden in Application
  - Use a direct access file (F format) similar to main memory
  - Implement a tree
    - use your favorite implementation
    - Main memory has also something like F format

Problems with this approach:
- Different implementations / compatibility
- Implementing records above of records...
Implementation with a Tree

- Which tree algorithm fits best for use with disks?
  - B tree
    - 2-3 trees
    - red-black trees
    - AVL trees
    - ...
  - B* tree / B+ tree
  - full tree or tree with data only in leaves

perhaps its a matter of taste (as always...)
Design of ISAM

ISAM = Index Sequential Access Method

- Design decisions done after properties of disks
  - Read and Write with disks is expensive (needs some milliseconds)
    - => n-ary tree, not a binary tree
    - => Size of a tree node is a disk block
  - Last level blocks are filled with records
    - F or V format, typically blocked
  - Only key in the index nodes
    - Update of records does not need the update of index node
Structure of an ISAM dataset
ISAM

- Inner nodes contain only key and block pointer
- Leave nodes contain records including the key
- V records are managed in the leaf block
  - F.e. an index for the start of each record
  - Moving records around in a block is fast, because the whole block is read and written
- Reorganization is delayed, because
  - Moving records between nodes needs more disk operations than a simple update
    - Combining 2 nodes, when fit into one node
    - Deletion when empty
  - Sometimes overflow lists to avoid reorg
ISAM Variants

• IBM Mainframe
  - Key and data are 2 different datasets
  - Different disks for speed

• Other OSes (BS 2000, VMS, …)
  - Key and data are in a combined dataset

• Unix
  - ISAM not available as part of the OS
  - ISAM not in the filesystem
    (at least not visible to user or applications)
ISAM under the cover

• C-ISAM
  - An ISAM implementation in C
  - Several variants available
  - Used in applications, where data with keys is necessary

• Databases
  - Most databases use typically ISAM structures
  - For primary and secondary keys
  - Good behaviour regarding number of IOs
Hash Index

• A hash function $h(key)$ returns an index
• The record is stored in a table at this position
  - F* format required

• Optimization:
  - Use a Disk block to store multiple records with the same hash value
  - Search on this block can be done sequentially
    • Much faster than reading from disk
    • Looking for one record needs also one read
Hash Index

- Hash function conflict:
  Different keys have the same hash value
  - Use buckets for records with the same key (f.e. blocks)
  - Rehashing
    - Apply the function again (or a different)
    - *(sorted) (minimal) perfect hash*
      - Modify the function for no conflicts
      - *(O(n) is possible, see: I. Wegener)*
  - Use *overflow lists*
Bitmap Index

- Requires: a low number of different values
  - low cardinality

- For each index and value
  - Bit list (1: has this value; 0: else)
  - Queries by logical operations on bit lists
    - using and / or / not logical operations
  - Sometimes: bit list for exception value
  - Optimization with run length encoding
    - needs logical operations for this data structure
Bitmap Index

- Can save space (low cardinality)
- Helps a lot
  in read-only databases
- Further speedup through
  - multiple fields also available as bitmap index
    - (department("HR") and status("manager")
      requires only an and operation
- Use in
  - Data Warehouses (Data Marts)