Organizations for key search

- Goal: Quick search for a record of a table with a given key value
Organizations for key search

- Definition. An organization is called *primary* if it determines the way records are physically stored, otherwise it is called *secondary*. 
Static and dynamic organizations

• Static:
  – After insertions or deletions may need a reorganization

• Dynamic:
  – Gradually evolves with insertion and deletions
Static hashing organizations

- Assumption: N records, with same and fixed size, stored in M pages of capacity c.
- Design Parameters
  - Page capacity (c)
  - Loading factor \(d = N/(M\times c)\)
  - Hash function H
  - Overflow management
Hash function

• Produces addresses uniformly distributed in the interval \((0, M-1)\)

• The typical hash function, with \(M\) prime:
  • \(H(k) = f(k) \mod M\)

• Two keys produce a collision if \(H(k_1) = H(k_2)\)

• If the number of collisions is greater than the page capacity, there is an overflow. Overflows increase the search cost.
Overflow management

• Open overflow:
  – Overflow records are put in the first available page

• Chained overflow:
  – Overflow records are put in a separate area, and chained together (by page or by record)
High loading factor reduces primary area size but increases the probability of overflow.
Page capacity

- If page capacity increases, overflows decrease.
- Hashing is convenient with a large page capacity ($\geq 10$).
Performance

• With few overflows:
  – Excellent performance for equality search
  – Range search?

• With many overflows:
  – Reorganization is needed
Dynamic hashing organizations

- With auxiliary data structures:
  - Virtual hash
  - Extendible hash

- Without auxiliary data structures:
  - Linear hash
  - Spiral hash
Virtual hash

Start with $H_j(k) = k \mod (2^j \times M)$, $j = 0$

$$H_j(3820) = 5$$

$$H_{j+1}(k) = H_j(k)$$

$$H_{j+1}(k) = H_j(k) + 2^j M$$
Virtual hash

PageSearch \((r, k)\):

\[ \textbf{if } B(H_r(k)) = 1 \textbf{ then } H_r(k) \]

\[ \textbf{else } \text{PageSearch } (r - 1, k) \]
Extendible hash

• Idea: substitute B with an index with references to pages, and double the index
• The index is smaller than primary area, and can be compressed with several techniques
Linear hash

Start with \( M \) pages and \( H_0(k) = k \mod M \quad H_i(k) = k \mod (2^i \times M) \)
Linear hash

\[ H_0(k) = k \mod M \]
\[ H_i(k) = k \mod 2^i \times M \]

function SearchPage(p, k: integer): integer;
begin
    if \( H_i(k) < p \)
    then SearchPage := \( H_{i+1}(k) \)
    else SearchPage := \( H_i(k) \)
end
Spiral hash

• With Linear Hash, unsplitted pages are crammed
• Better a spiral data area
• The hashing function is uneven. The load is high at the beginning of the address space and tapers off towards the end.
Summary

• Hash organizations are simple to implement

• Static
  – When is well designed (80% page occupancy), a record is retrieved with ~1 page access
  – Hard to keep 80% when size changes

• Dynamic
  – Complex but well behaved (spiral hashing)

• Problem:
  – for range search they are useless!