PRINCIPLES OF DATA ORGANISATION

Fagin

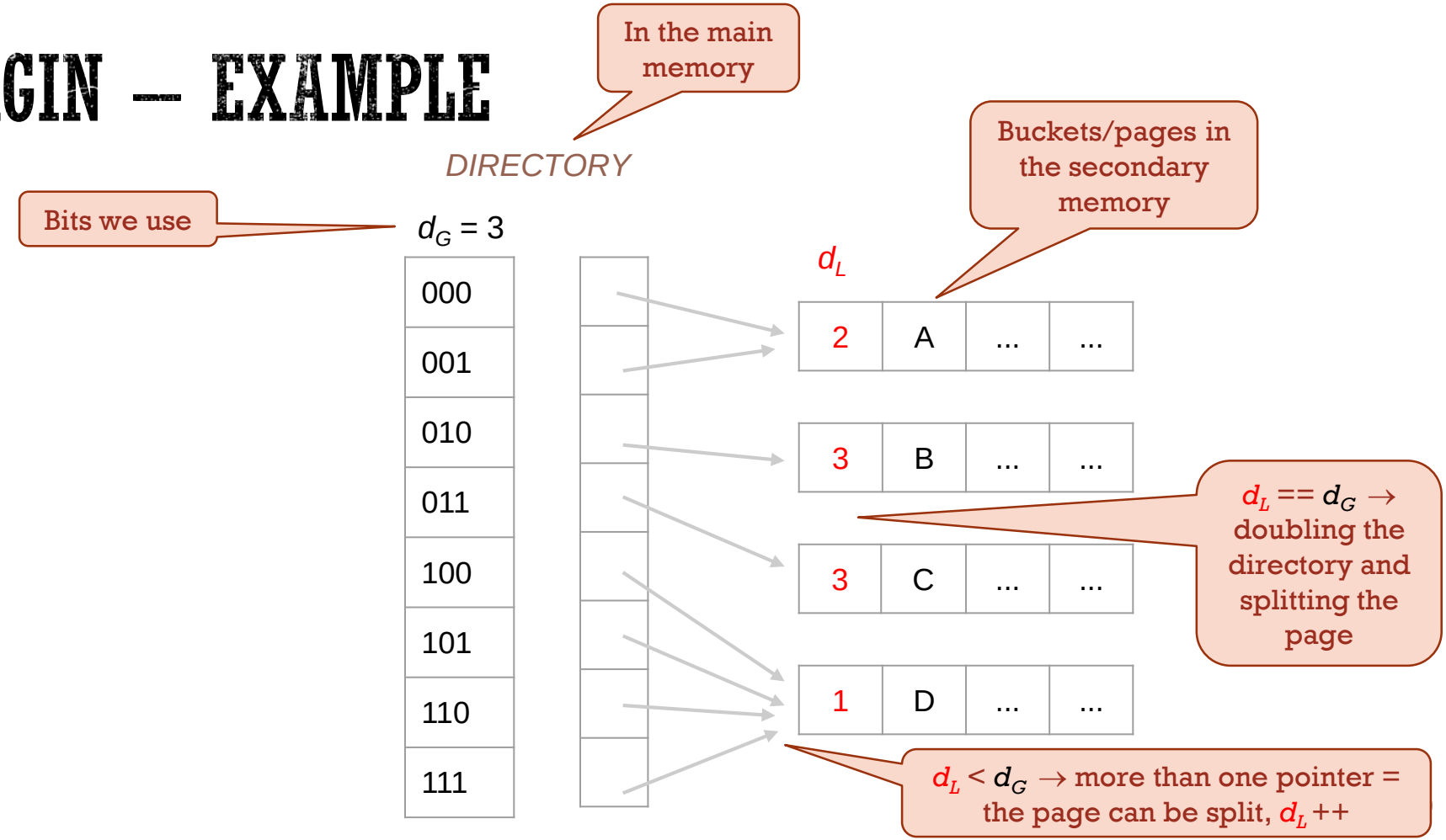
MOTIVATION

- Key, pointer pairs ~ index
- Hashed file organization
- Dynamic hashing
  - We can add new records (without performace penalty)
  - We do not need to specify the amout of data beforehand
- Collapsing a trie (prefix tree)
  - We use a growing part of the tree

FAGIN'S EXTENDIBLE HASHING

- Fagin 1979
- In general: hash function ***h(k)*** returns a string of bits
  - But we do not need all of them all the time
- Directory based
  - Level of indirection = we do not need a continuous space in the secondary memory
  - Global depth d_G
    Bits needed to tell any pair of records from different buckets apart
  - Local depth d_L (own for each bucket/page)
    Number of bits common to all records in a bucket
    - $2^{(d_G - d_L)}$ = how many directory records point to a page
- Hash function (uniform, fast,…) provides d_G-long address of the directory entry with a pointer to the bucket/page
- Overflowing causes a change in the structure of the directory (d_g, d_L) and the primary file
  - Adding new blocks or modyfying (doublong) the directory

FAGIN – EXAMPLE



FAGIN – FIND

Finding a record with a key k

- Compute $k' = H(k)$
- Compute $k'' = h_{d_G}(k')$
- Access page pointed to by the directory record with key k''
- Scan the accessed page for record with key k
  - If the record is not found, it is not present in the file

FAGIN – INSERT

Inserting a record R with a key *k*

- Find a page where the record R should be inserted
- If the page is not full, insert R into the page and return
- If the page is full, split the page
  - Locally
  - Globally

FAGIN – SPLIT LOCAL

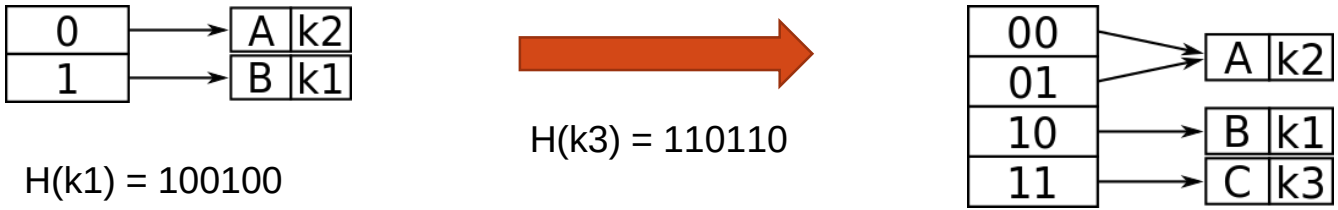
Split page ***P*** if $d_L(P) < d_G$

- Allocate new page ***Q***
- Modify the directory pointers originally pointing to ***P*** so that, e.g., half of them having common first $d_L(P)$ bits followed by **0** point to ***P*** and rest of them point to ***Q***
- Set $d_L(P) = d_L(Q) = d_L(P) + 1$
- Reinsert all the data from ***P***

FAGIN – SPLIT GLOBAL

Split page ***P*** if $d_L(P) == d_G$

- Double the directory size
- $d_G = d_G + 1$
- For each page ***Q***, set the pointers so that if ***Q*** was pointed to by an entry with a bit key ***x***, now it is pointed by entries with keys starting with ***x0*** and ***x1***



FAGIN

- The performance stays more or less constant with increasing number of stored records
- The directory might not fit in the main memory
- If the block factor is low, many splits can occur leaving many pages empty