HASH INDEXES

CS 564- Fall 2021

ACKs: *Dan Suciu, Jignesh Patel, AnHai Doan*

WHAT IS THIS LECTURE ABOUT?

Hash indexes

- Static Hashing
  - I/O cost
  - issues with static hashing
- Extendible Hashing
  - insertion
  - deletion

HASH INDEXES

- efficient for equality search
- not appropriate for range search

- Types of hash indexes:
  - static hashing
  - extendible (dynamic) hashing

STATIC HASHING

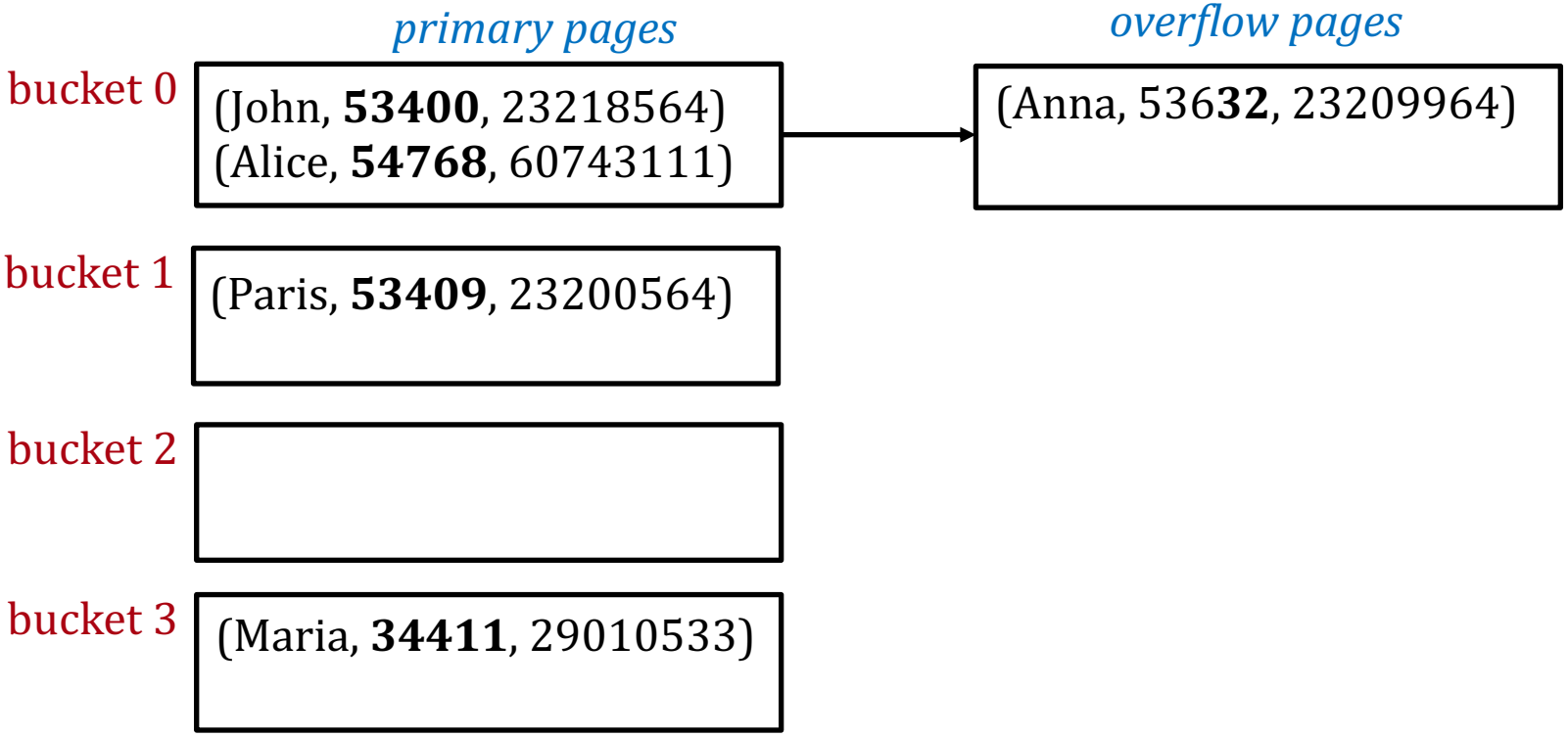
- A **hash index** is a collection of *buckets*
  - bucket = primary page + overflow pages
  - each bucket contains one or more index entries
- To find the bucket for each record, apply a hash function ***h***
- ***h*** maps a search key value to one of the buckets

STATIC HASHING: EXAMPLE

`Person(name,zipcode,phone)`

- *search key*: {zipcode}
- *hash function* ***h***: zipcode mod 4

- 4 buckets
- each bucket holds 2 index entries



OPERATIONS ON HASH INDEXES

Equality search (*search-key = value*)

- apply the hash function on the search key to locate the appropriate bucket
- search through the primary page + overflow pages to find the record(s)

I/O cost = 1 + #overflow pages

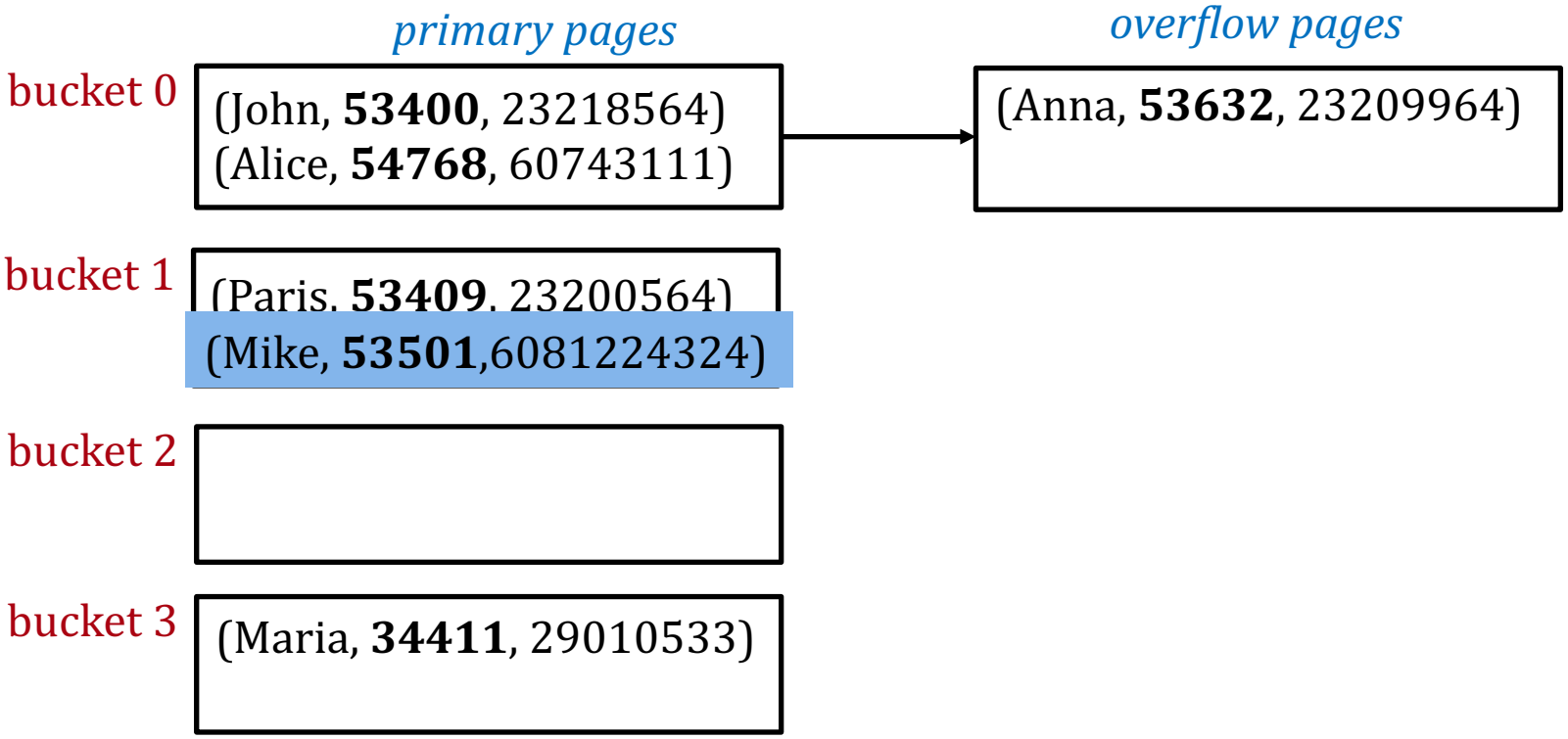
The I/O cost increases if we choose alternative #2

OPERATIONS ON HASH INDEXES

- **Deletion**
  - find the appropriate bucket, delete the record
- **Insertion**
  - find the appropriate bucket, insert the record
  - if there is no space, create a new overflow page

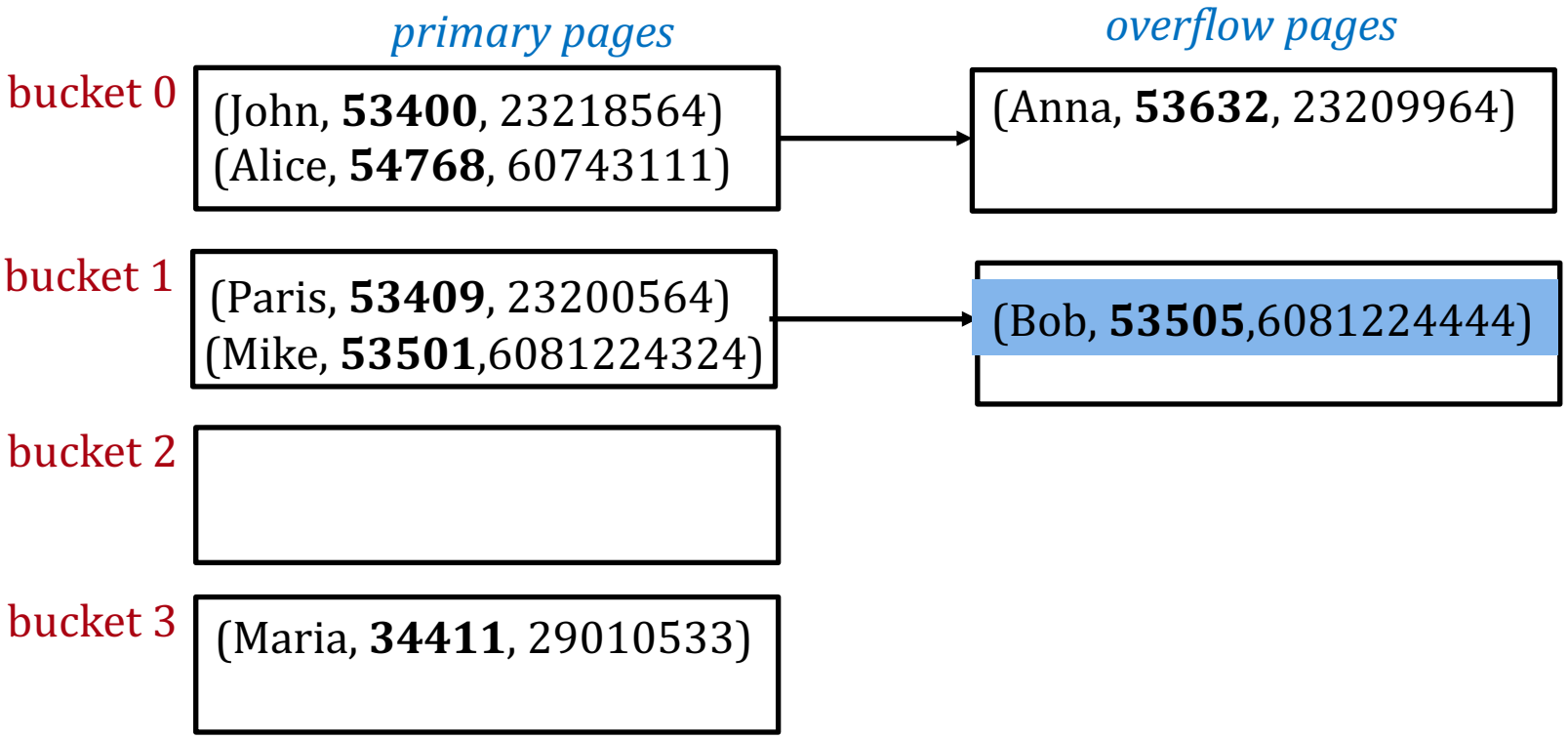
STATIC HASHING: INSERT

insert: (Mike, **53501**,6081224324)



STATIC HASHING: INSERT

insert: (Bob, **53505**,6081224444)



HASH FUNCTIONS

- An *ideal* hash function is uniform: each bucket is assigned the same number of key values
- A *bad* hash function maps all search key values to the same bucket
- Hash functions in practice:
  - should be fast to compute
  - should have a low collision rate

PROBLEMS OF STATIC HASHING

- In static hashing, there is a **fixed** number of buckets in the index
- Issues with this:
  - if the database grows, the number of buckets will be too small: long overflow chains degrade performance
  - if the database shrinks, space is wasted because of empty buckets
  - reorganizing the index is expensive and can block query execution

PROBLEMS OF STATIC HASHING

- Even with a good hash function, long overflow chains can still occur when we have **skew**
- Skew occurs when many records have the same search key value, so they will always end up in the same bucket

EXTENDIBLE HASHING

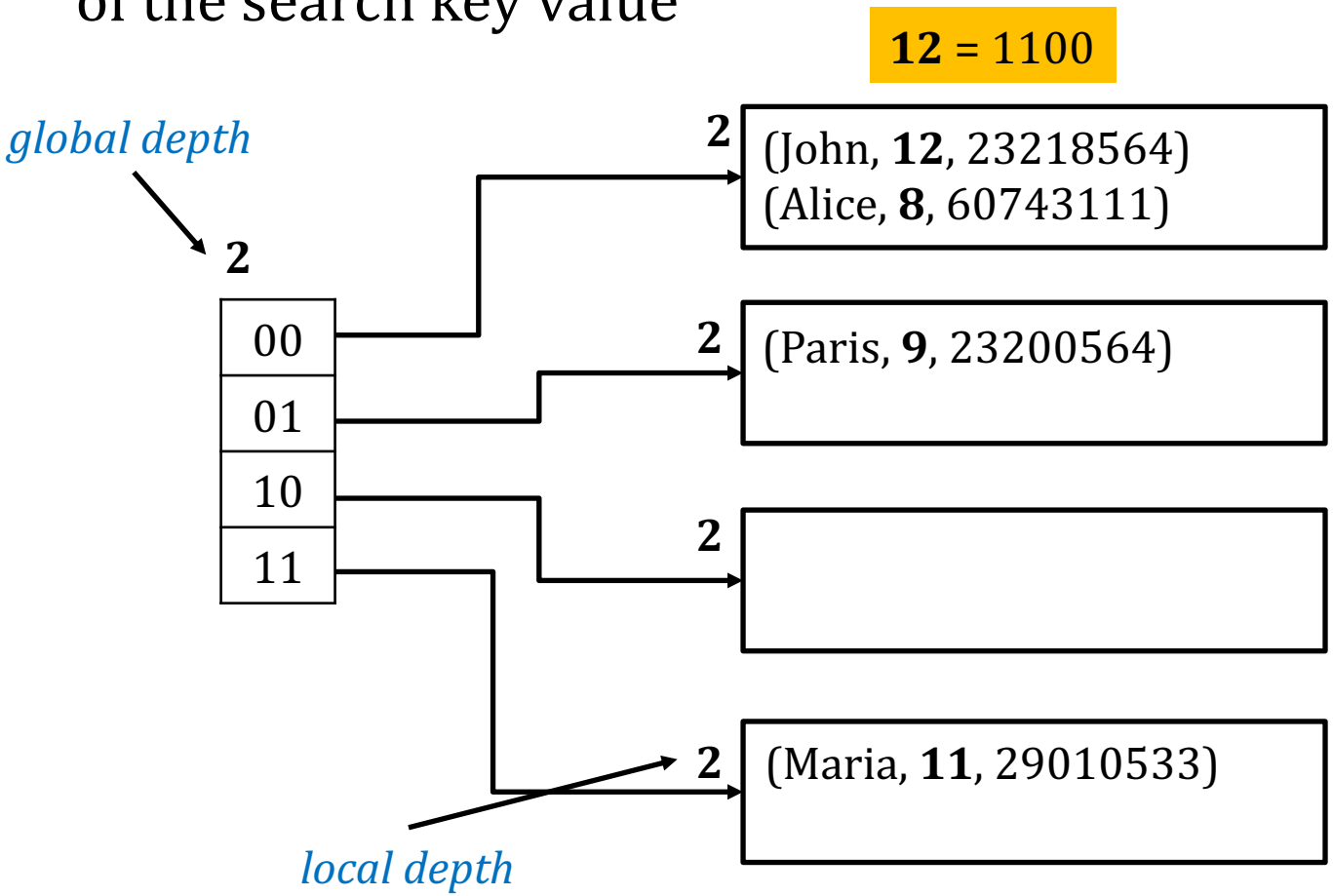
EXTENDIBLE HASHING

Extendible hashing is a type of *dynamic* hashing

- It keeps a directory of pointers to buckets
- On overflow, it reorganizes the index by doubling the directory (and not the number of buckets)

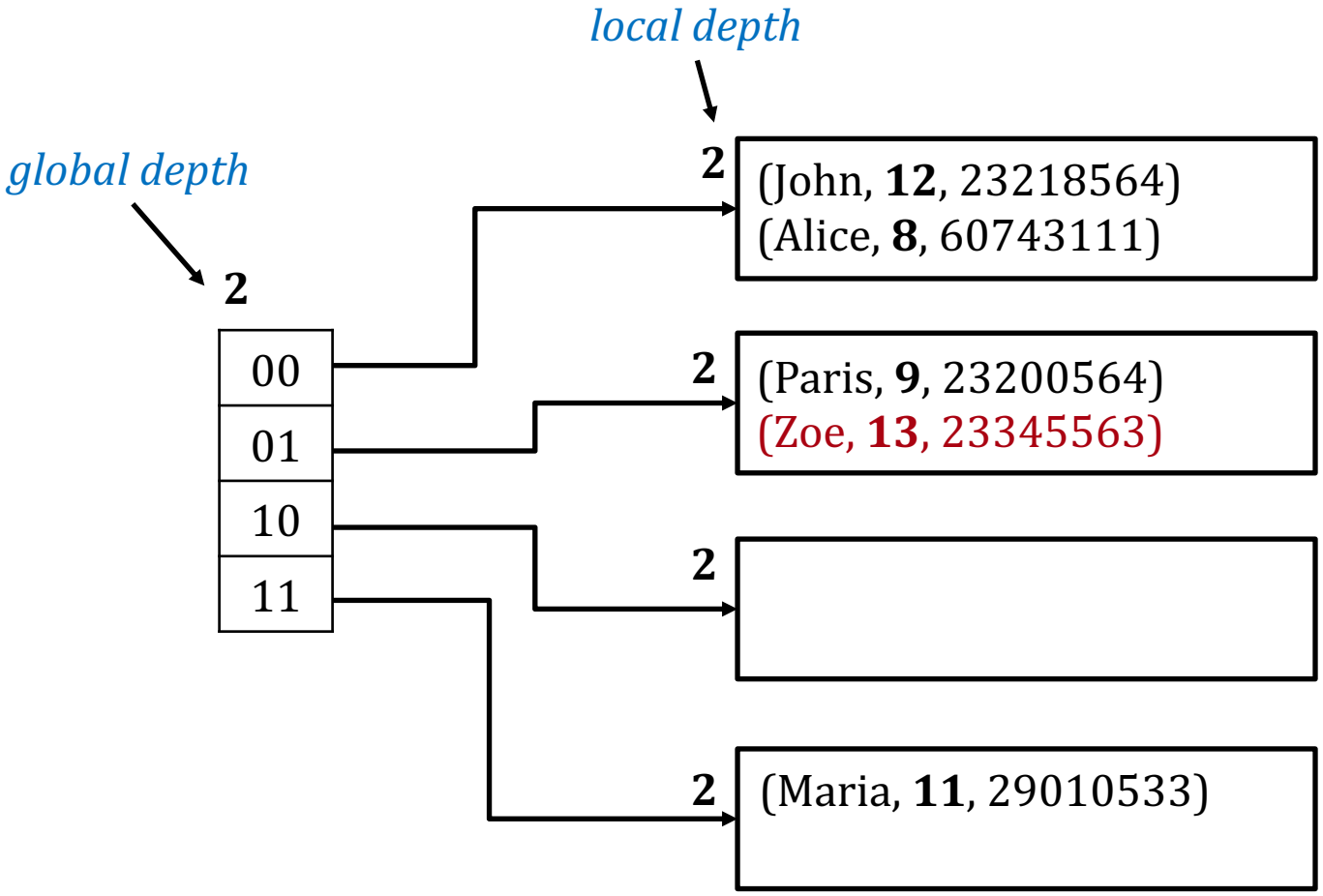
EXTENDIBLE HASHING

To search, use the last *(global depth)* digits of the **binary** form of the search key value



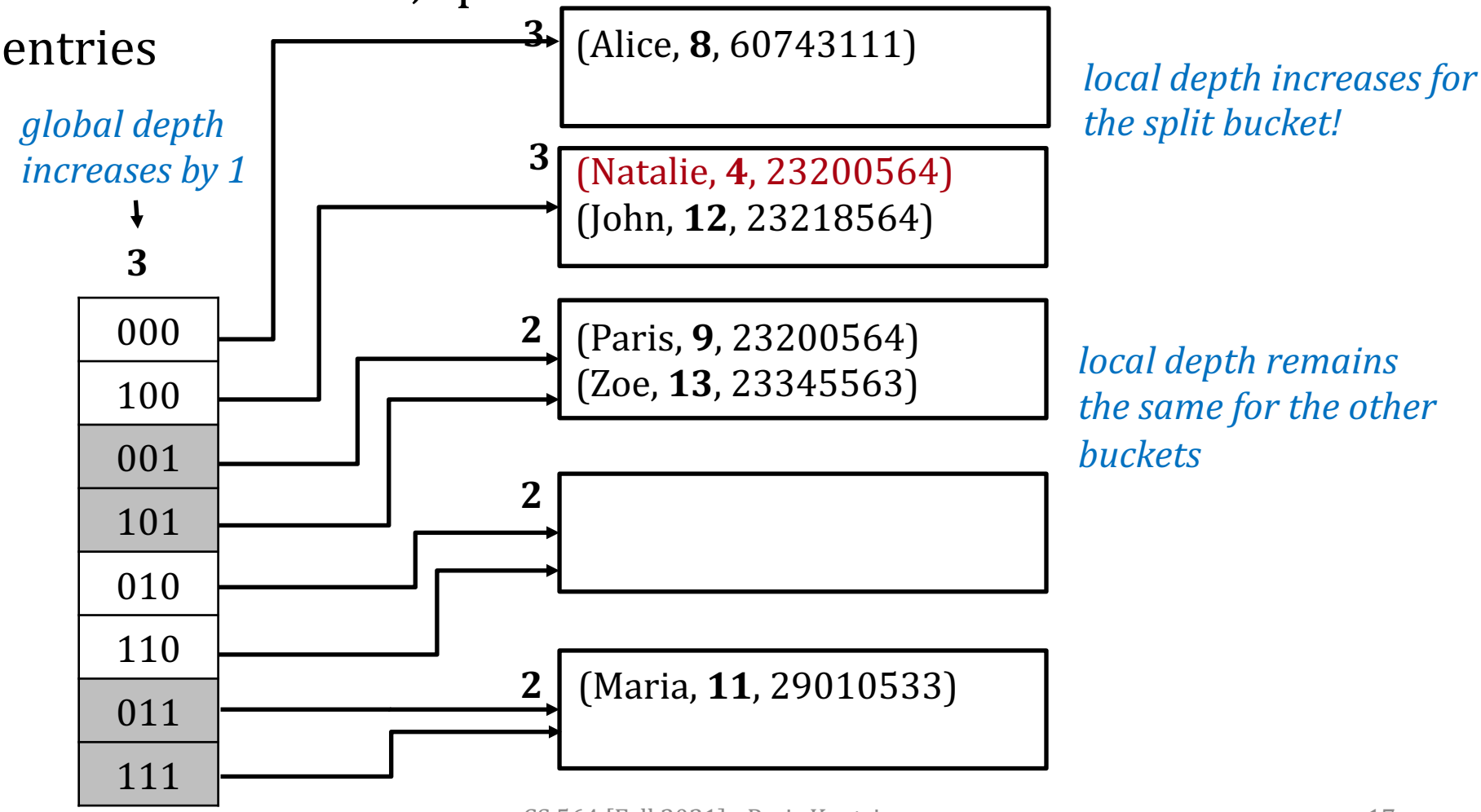
EXTENDIBLE HASHING: INSERT

If there is space in the bucket, simply add the record



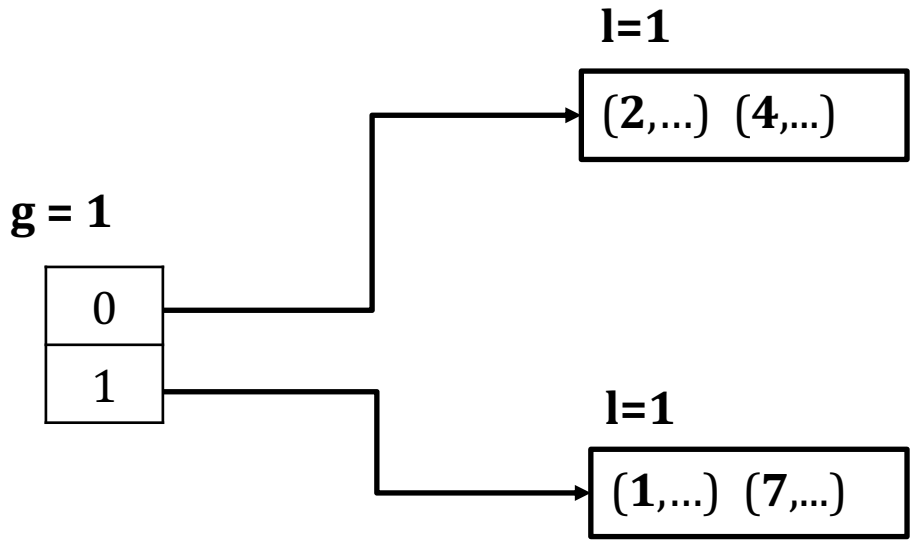
EXTENDIBLE HASHING: INSERT

If the bucket is full, split the bucket and redistribute the entries



EXAMPLE

each page can hold at most two records

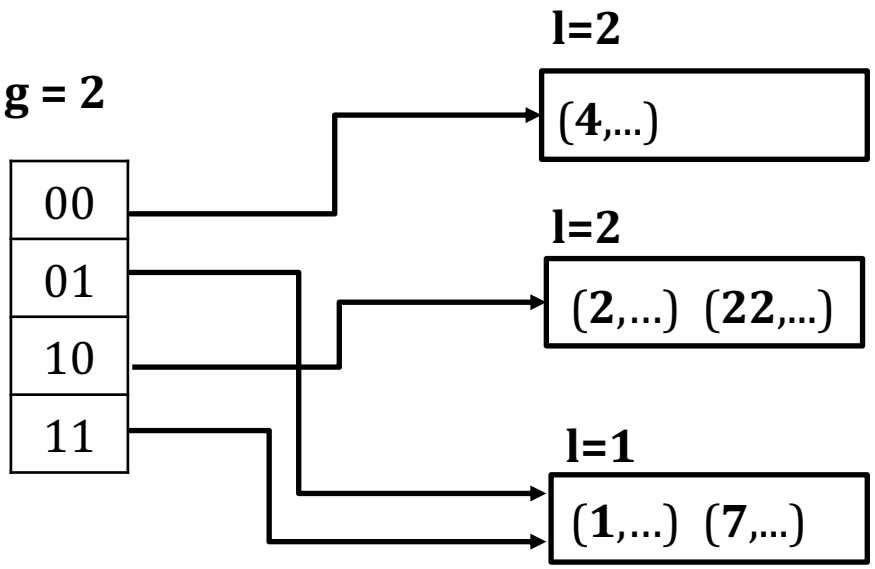


INVARIANT: global depth >= local depth

EXAMPLE

- The catalog doubles in size
- Global depth becomes 2

insert: (**22**,…)



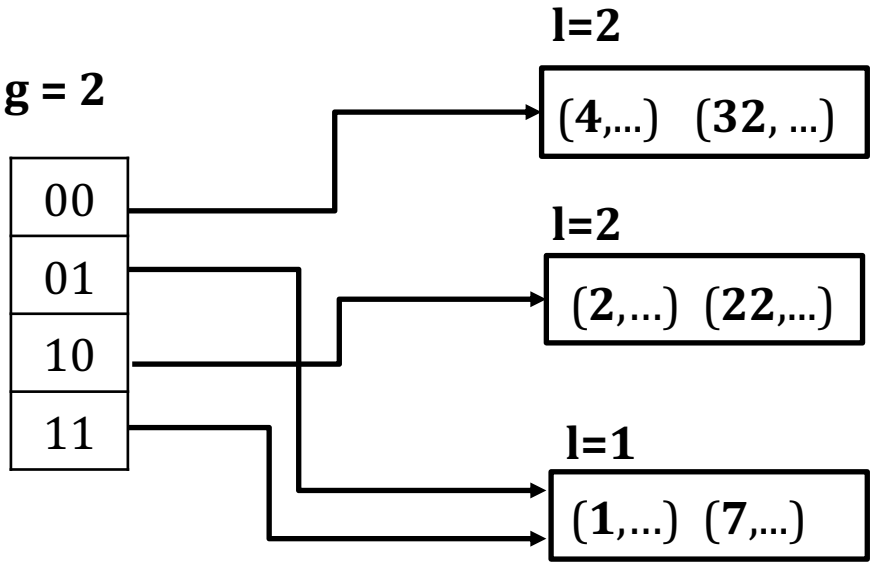
This bucket is split into two buckets with local depth 2

This bucket remains the same

EXAMPLE

There is space in the bucket so nothing changes!

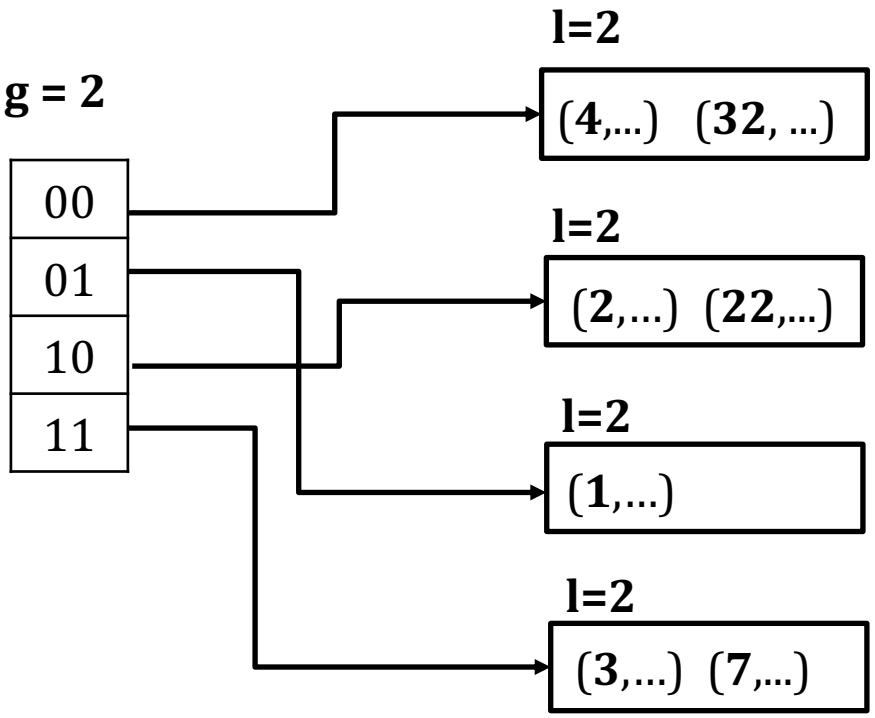
insert: (**32**,…)



EXAMPLE

Since local depth is smaller than global, no need to change the directory size!

insert: (3,…)



The bucket is split into two buckets with local depth 2

EXTENDIBLE HASHING: DELETE

- Locate the bucket of the record and remove it
- If the bucket becomes empty, it can be removed (and update the directory)
- Two buckets can also be coalesced together if the sum of the entries fit in a single bucket
- Decreasing the size of the directory can also be done, but it is expensive

MORE ON EXTENDIBLE HASHING

- How many I/Os for equality search?
  - One if directory fits in memory, else two
- The directory grows in spurts, and, if the distribution of hash values is skewed, the directory can grow very large
- We may need overflow pages when multiple entries have the same hash value!