DBT2 Exercise
Idx Solutions

TECHNISCHE UNIVERSITÄT DARMSTADT

Robert Gottstein

gottstein@dvs.tu-darmstadt.de

Why Indexing?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Accesspath To IO Subsystem
- No Need to search on the data itself
  - Sometimes the Data is ordered, sometimes not
    - Heap, Sorted, Hashed
  - Clustered Data
  - Clustered Index (Idx+Data altogether)

- Difference between a Search and a Seek?
  - Is faster to search for a value or to seek a position korrelated with a value?

Indexing – Primary Index vs. Primary Key

TECHNISCHE UNIVERSITÄT DARMSTADT

„If the file containing the records is sequentially ordered, a **clustering index** is an index whose search key also defines the sequential order of the file.

Clustering indices are also called **primary indices**;

the term primary index may appear to denote an index on a primary key, but such indices can in fact be built on any search key.“

The search key of a clustering index is ***often*** the primary key, although that is not necessarily so.

Indices whose search key specifies an order different from the sequential order of the file are called **nonclustering indices** or **secondary indices**.

p.476 Chap. 11.2 Database System Concepts – A. Silberschatz

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

1. What is the main difference between a B and a B+ Tree?

   B+ Tree has a seperate Index and Data Part. The leaves in a B+ Tree are the data Items while the branch nodes are pointers and contain no data. The B+ Tree therefore has a higher Fan-Out than the B-Tree.

2. Can you make an example where the access depth in a B+ of two values is different (Different High)?

   No! The height is the same for each value.

3. Define the Criteria of a minimum/ maximum filled B+Tree
   1. Each path from root to leaf has the same length
   2. Each node but the root and the leaf has min. k+1 successors. The root has at least 2 successors exept it is a leaf.
   3. Each inner node has at max. 2k+1 successors
   4. Each leaf node (but the root iff it is a leaf) has a least k and at max 2k entries

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

4. Given a B-tree of class T(8,13), what are its maximal and minimal number of keys? Justify your answer.

The Basics (must know!):

*To get the number of **keys** – you first need to know how many **nodes** there are in the tree!*

*The **root** is a node that contains 1 Key (at least)*
*How many **sons** does it therefore have??*

Branches are nodes which contain at least k keys and at max 2k keys
How many sons can they have??

To know how many keys are in the B+ Tree, do you count the leaf nodes??

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

4. Given a B-tree of class T(8,13), what are its maximal and minimal number of keys? Justify your answer.

Min # of keys in a B-Tree $2 \cdot (k+1)^{h-1} - 1$

Max # of keys in a B-Tree $(2k+1)^{h} - 1$

Grundlagen der Informatik II
P.162 Härder GDI2
Formula 7.1; 7.2; 7.4; 7.5

nodes (min) $1 + \frac{2}{k} \cdot ((k+1)^{h-1} - 1)$

#nodes (max) $\frac{(2k+1)^{h} - 1}{2k}$

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

4. Given a B-tree of class T(8,13), what are its maximal and minimal number of keys? Justify your answer.

For a B-tree B of class $\tau(k,h)$, each node (except the root) contains at least k keys. In order to obtain the *minimum* number of keys of B, its root must contain only one key, thus point to two subtrees of order $k+1$ and height $h-1$. In order to obtain the *maximum* number of keys, each node may have up to $2k$ keys, i.e., have order $2k+1$.

The number of nodes a tree B has, $N(B)$, can be obtained as follows:

$$1 + \frac{2}{k} * \left((k+1)^{h-1} - 1\right) \leq N(B) \leq \frac{(2k+1)^h - 1}{2k}$$
$$122,070,313 \leq N(B) \leq 6.19036127 \cdot 10^{14}$$

p.162 Formula 7.3

The number of keys n can thus be derived:
$$2 \cdot (k+1)^{h-1} - 1 \leq n \leq (2k+1)^h - 1$$
$$564,859,072,961 \leq n \leq 9.90457803 \cdot 10^{15}$$

Therefore, a B-tree of class $\tau(8,13)$ can contain minimally $564,859,072,961$ and maximally $9.90457803 \cdot 10^{15}$ keys.

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

5. Describe how a key set can be inserted into a B-tree of class T(k, h) such that the number of nodes becomes maximal.

In order to achieve a maximum number of nodes in a B-tree of class T(k, h), we must insert the keys in a **sorted order**. If we insert the keys in ascending sorted order, the new key is always placed on the node at the right bottom. If the node is full, it is split into two nodes which both are minimally occupied now. The median key is moved into the parent node. This process will lead to a B-tree with a maximal number of nodes when executed.

6. How should the B-tree look like to make writing operations faster?

The cost for writing accesses is essentially influenced by the need for **reorganization**. The tree should therefore be as **sparsely** populated as possible. Alternatively, a **larger split factor** m can be used, or the split operations can be postponed

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

7. How can you speed up the construction of a B-tree for a given, large set of keys?

Through **bulk loading**. Instead of inserting every key one by one, we **first sort the set** of keys with an efficient sorting algorithm, create an empty multi-way tree and then insert the keys, node per node, such that it results in a B-tree.

1. B/B+ - Tree Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

8. What are the advantages/disadvantages of B-Trees?

- Dynamic index:
  - **overflow** requires node **splitting**
  - **underflow** requires node **merging/rebalancing**
  - splitting may go up all the way to root
- Entries in leaf nodes are index value/tuple ID pairs
  - **optimization** possible through single value header followed by TID list
  - each TID requires 4 Bytes, header neglectable with long lists of TIDs
- **Chaining** of leaf nodes for range processing
- In OLTP environment fill usually ~ 66% → 75% depending on key rotation
- With many tuples and small domains, single index value followed by long lists of tuple IDs

Bitmap Index

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Calculate the size of the Bitmap Index: The Index is on the column „colour“ of the table „subpixel“. The pixel is constructed out of 3 possible subpixel (red, green, blue). The table indexes a Full HD Display.

You will need 3 Bitmap Vectors. Full HD is 1920*1080P which are 2073600 pixels = amount of tuples. Each Pixel needs 3 Subpixels = 3 Bitmap Vectors.
→ 254Kb per Vector → * 3 = 762Kb

2. How to maintain a bitmap Index? (Expansion of Domain, No Expansion of Domain)
Maintaining bit map indices is easy
No expansion of domain: add 1 in last position to vector corresponding to value of attribute in new tuple
With expansion of domain: add new vector with leading 0s and 1 in last position, add 0 to all other vectors.

Bitmap Index

TECHNISCHE UNIVERSITÄT DARMSTADT

3. How to scan a Bitmap Indexed Table?

Bit map indices are scanned, **position in bit vector corresponds to position of tuple in table**

```
SELECT *
FROM SUPPLIER S, SUPPLY SP
WHERE S.Quality > 5.0 AND
      S.SID = SP.SID AND
      SP.ITEM IN {a,b,c}
```

Selection predicate can be evaluated directly on bit-map index by **ANDing** the bit vectors for SP.ITEM=a, SP.ITEM=b and SP.ITEM=c

Bit-wise logical operations are fast!!

Bitmap Index

TECHNISCHE UNIVERSITÄT DARMSTADT

4. Which Operations can be performed on Bitmap Idx?

Bitmaps (if not kept in memory) are read into main memory in large fragments and easily **ANDed** (**ORed**) [All **Boolean** Operations]

```
for (i = 0); i < len(B1); i++)
   B3[i] = B1[i] & B2[i];
```

Count operations are usually performed on the foundset
If foundset is represented as bitmap, it can be easily parallelized (define a vector of counters - one for each fragment, count fragments in parallel, sum counters)

Bitmap Index

TECHNISCHE UNIVERSITÄT DARMSTADT

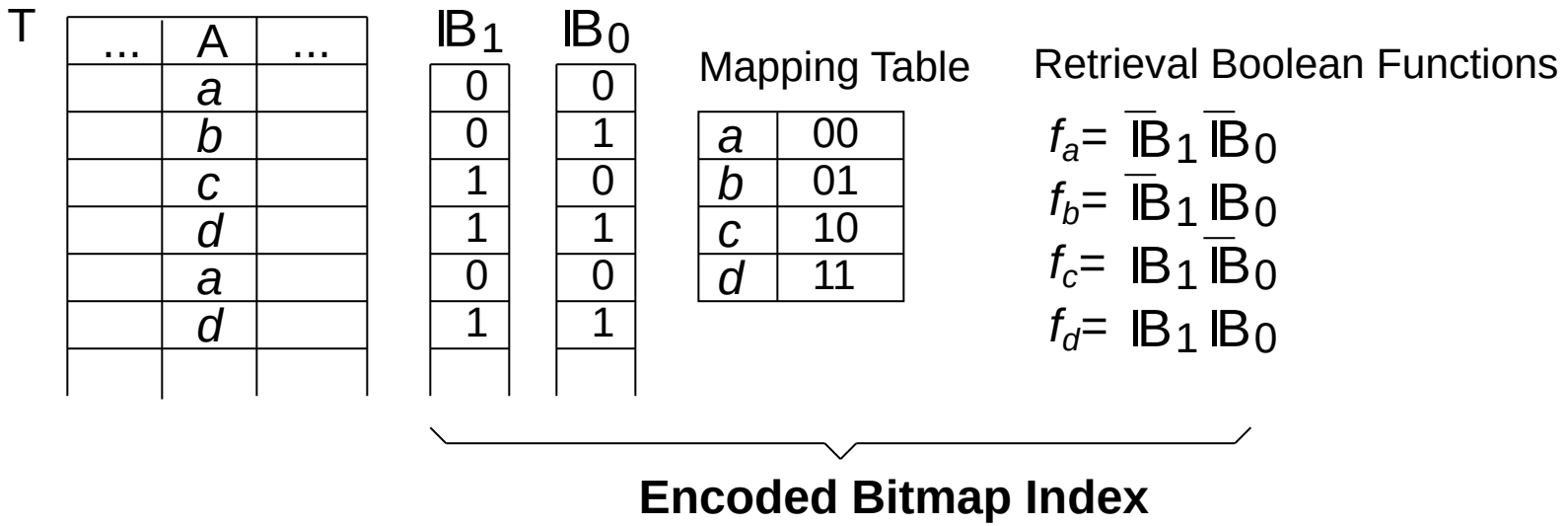
5. What are the (dis-)advantages of Bitmap Idx over traditional idx (B-Trees) and what are the limitations?

- Advantages of bitmap indexes [P.O'Neil] over traditional indexing (B-trees)
  - easy to construct, easy to maintain and easy to use
  - cooperativity of different indexes, i.e. logical operations in the WHERE-clauses can be simply evaluated by performing corresponding logical operations on the bitmaps
  - fast for selections involving logical operations (AND, OR, NOT) on indexed values
- Simple bitmap indexing is **unsuitable for large domains**
  - increasing sparsity
  - performance degrades
- Solutions
  - compression of simple bitmaps (run-length encoding)
  - encoding the domain of the indexed attribute

Bitmap Index

6. What is an encoded Bitmap Index? What problem(s) does is solve?

- An encoded bitmap index consists of a set of bitmap vectors (bit slices), a mapping table and a set of retrieval Boolean functions.



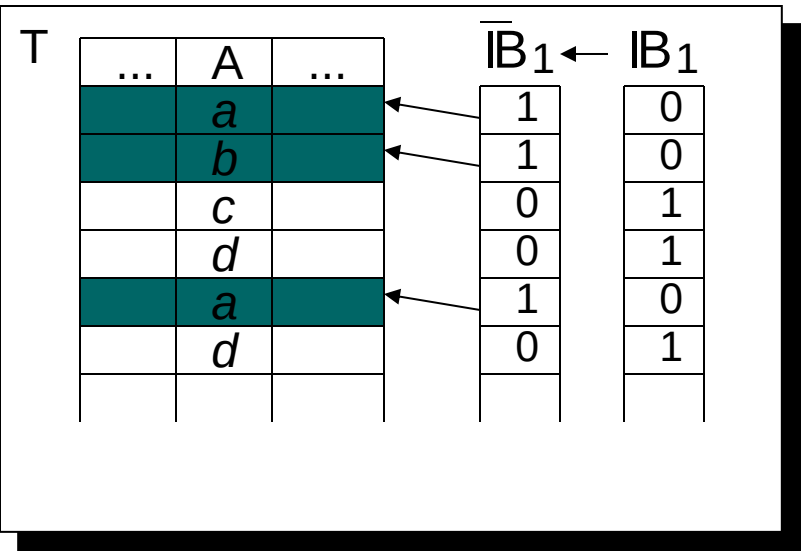
Bitmap Index - How EBI Works

```
SELECT *
FROM T
WHERE T.A in {a, b}
```

$f_a = \overline{B}_1 \overline{B}_0 \qquad f_b = \overline{B}_1 B_0$

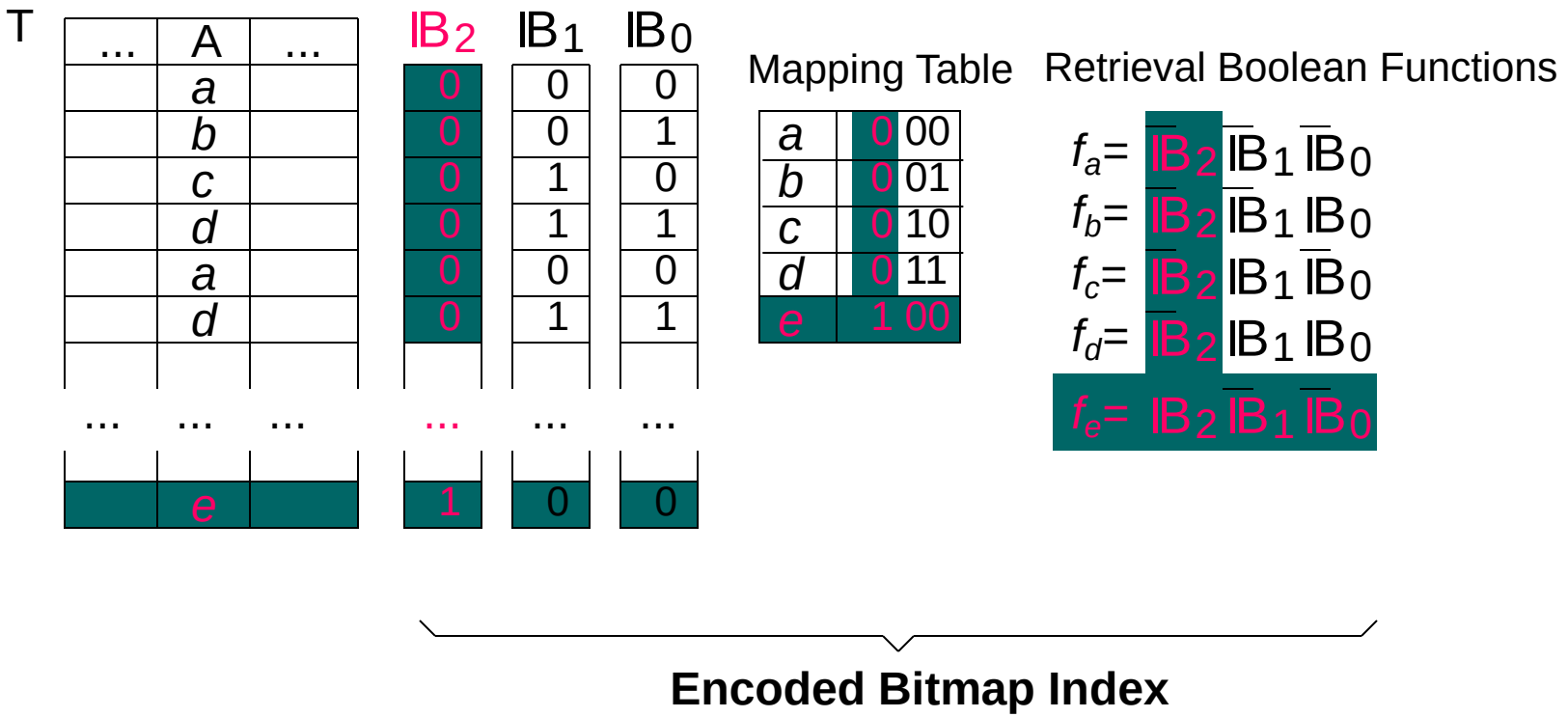
$f_a + f_b = \overline{B}_1 \overline{B}_0 \text{ OR } \overline{B}_1 B_0$

$= \overline{B}_1$



Maintaining EBI

- appending with domain expansion



Z – Order

Given is a 4 Bit Encoded 2D Space which is identified by pairs (y, x)

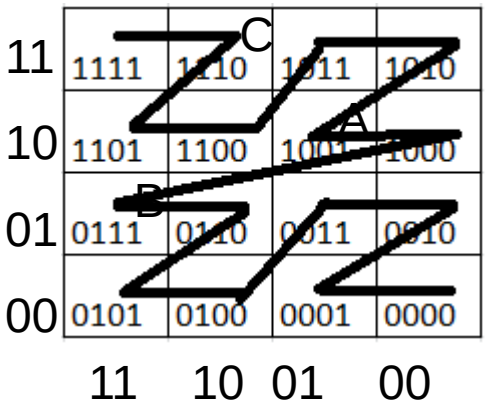
1. Create a Z Transformation on the given Space
2. How do you determine the adress of A, B, C via the Z-Order?

Via Bit Interleaving:

A: 10 ~ 01 → 1001

B: 01 ~ 11 → 0111

C: 11 ~ 10 → 1110



Z - Order

TECHNISCHE UNIVERSITÄT DARMSTADT

What are the advantages of the Z-Order?

- *Z*-order encoding preserves the spatial proximity of points
- homogeneous regions are represented compactly
- the elements are clustered => efficient access to secondary storage
- *Z*-order coded data can be stored into secondary storage using
- conventional prefix B+ trees
- efficient “range queries” are possible
- direct access via *z*-value

Indexing HDD vs. SSD

TECHNISCHE UNIVERSITÄT DARMSTADT

- What do you Think??
- HDD moves the Head (even with multiple platters the heads move as a bunch)
- SSD Multiple Chips which can be accessed in parallel

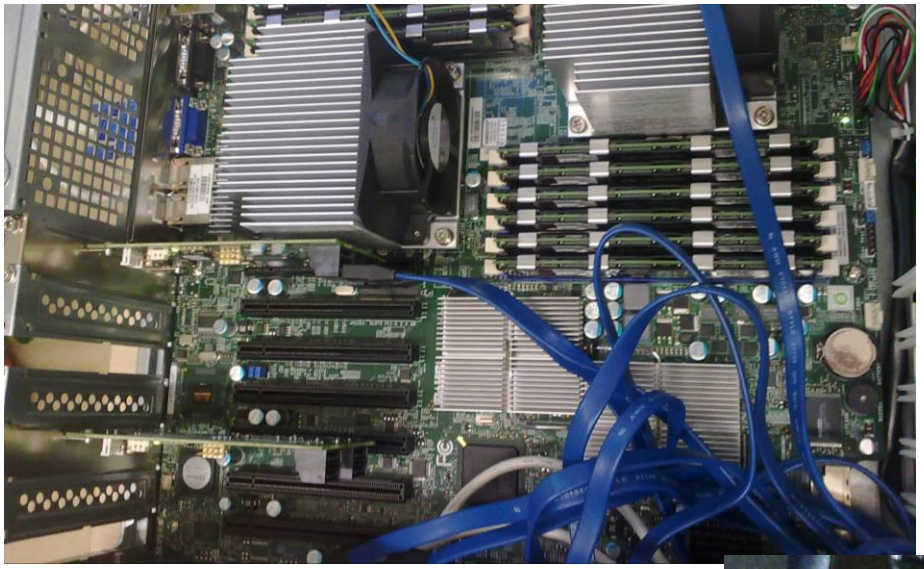
- Is it possible to traverse an index in parallel?
  - This would speedup the Access on an SSD!

- B.Sc. as well as M.Sc. Thesis available at Project FlashyDB
  - Just contact me

  - Have Access to new technology tesbeds
    - SSDs, Servers with 16 Core CPUs, Large Memories…

FlashyDB – Working on Flash

TECHNISCHE UNIVERSITÄT DARMSTADT



Sylt Server (one of two)

- 2xIntel (=16 Logical Cores)
- 48 GB Ram
- 2x LSI Controller
- **8 SSD (Up to 12)**
- …and a Bunch of HDDs

Working on new Technology
With known DB Systems

FlashyDB – Working on Flash

TECHNISCHE UNIVERSITÄT DARMSTADT

And Yet there is more…

- Sylt
- Rom

- Panamera
- Boxter
- Spyder
- Cayenne
- ….