Indexing techniques

Assoc. Prof. Dr. DANG Tran Khanh

khanh@hcmut.edu.vn

Outline

- Indexing in database systems
- Traditional indexing: B/B$^+$ tree
- Spatial indexing: R/R* tree
- Indexing techniques for high-dimensional data
  - Curse of dimensionality
- New indexing techniques for emerging application domains
- Exercises
- Reading: chapters 16, 17 [1] and internet

Indexing in database systems

- A way to optimize performance of a database by minimizing the number of disk accesses required when a query is processed
- Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed
  - **Primary Index**: defined on a data file that is ordered on a **key field**. The key field is generally the primary key of the relation
  - **Secondary Index**: generated from a field which is a candidate key or a non-key
  - **Clustering Index:** defined on a data file ordered on a **non-key field**
- Ordered Indexing has two types: **Dense** & **Sparse**

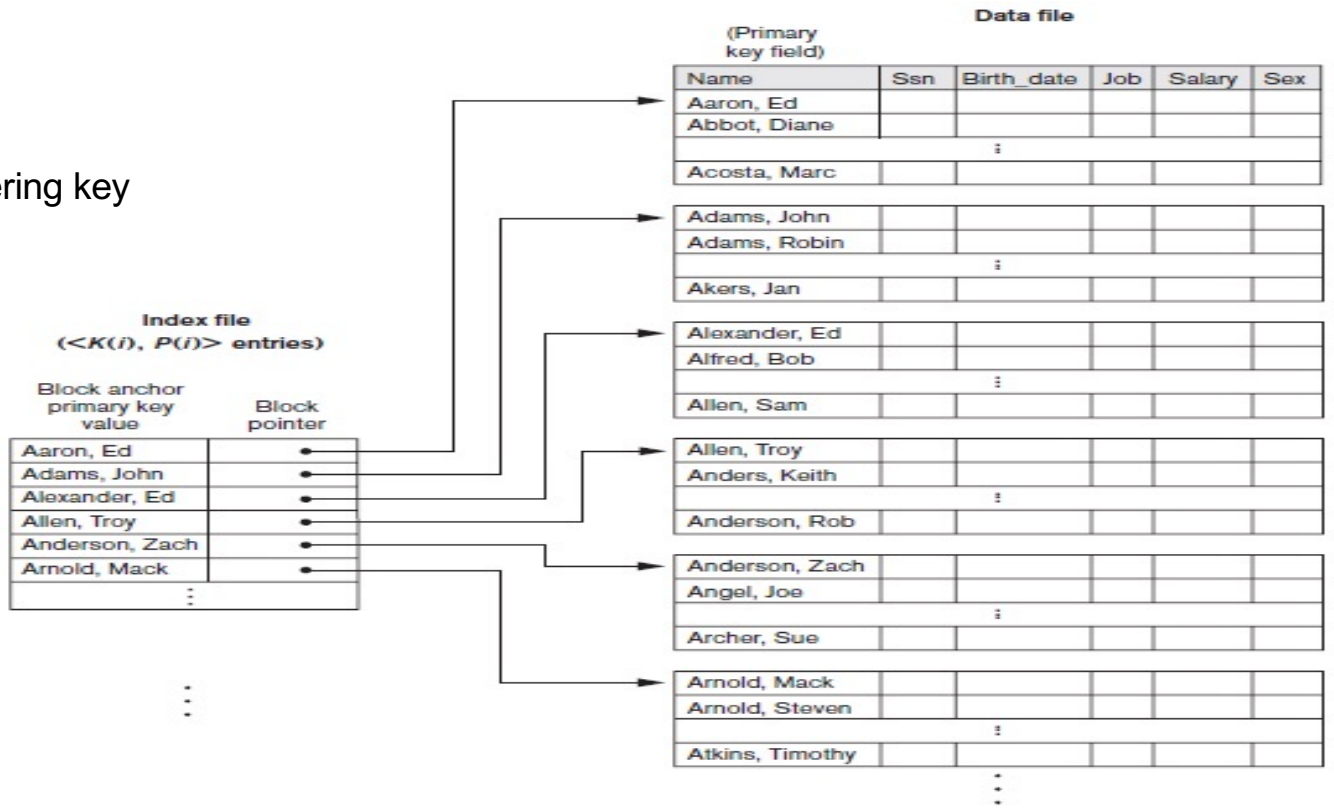
Indexing in database systems

- **Dense Index:** An index record appears for **every** search key value
  - This record contains search key value and a pointer to the actual record
- **Sparse Index:** Index records are created only for **some** of the records
  - To locate a record, find the index record with the largest search key value less than or equal to the search key value we are looking for
  - Start at that record pointed to by the index record, and proceed along the pointers in the file until we find the desired record

Indexing in database systems

- **Primary Index**: defined on a data file that is ordered on a **key field**. The key field is generally the primary key of the relation

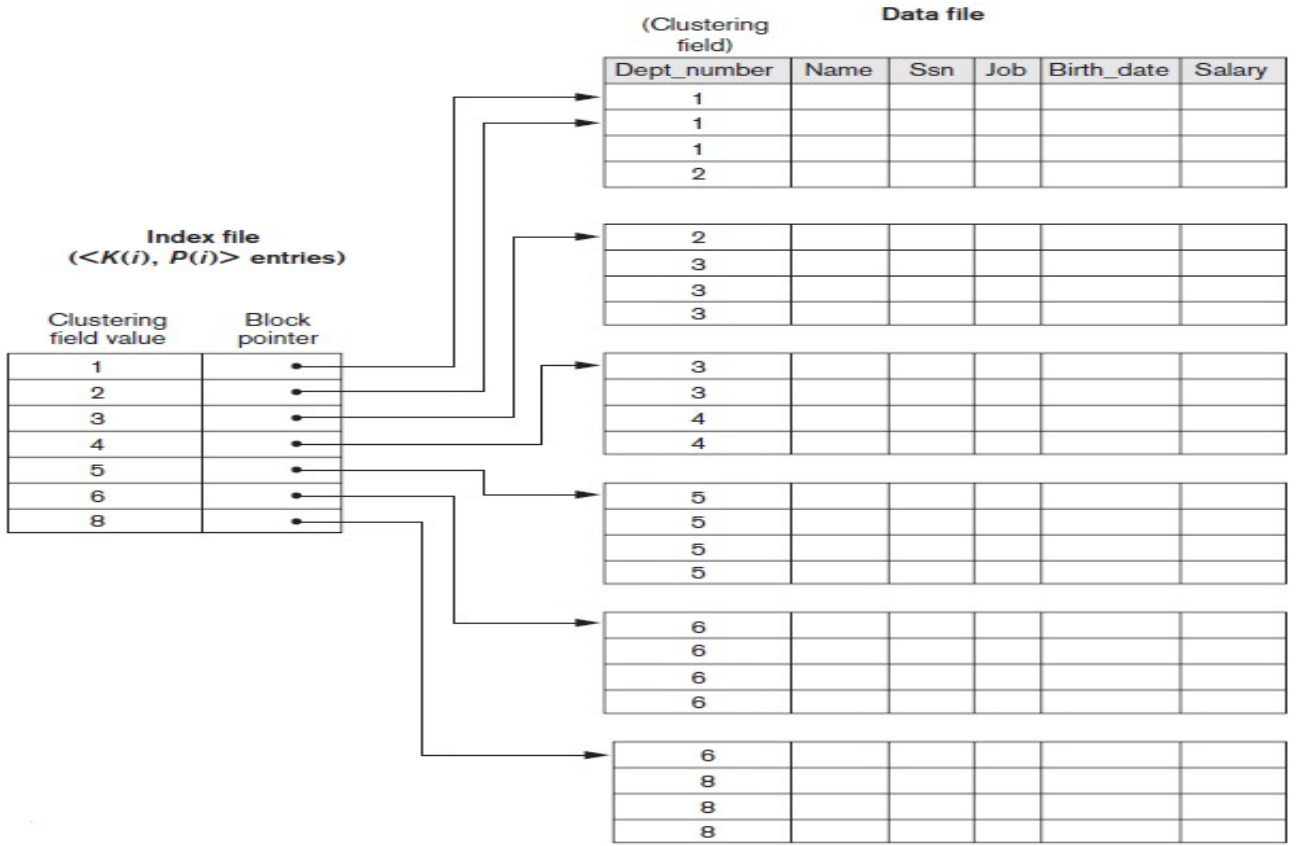
Primary index on the ordering key field of the file



Indexing in database systems

- **Clustering Index:** defined on a data file ordered on a **non-key field**

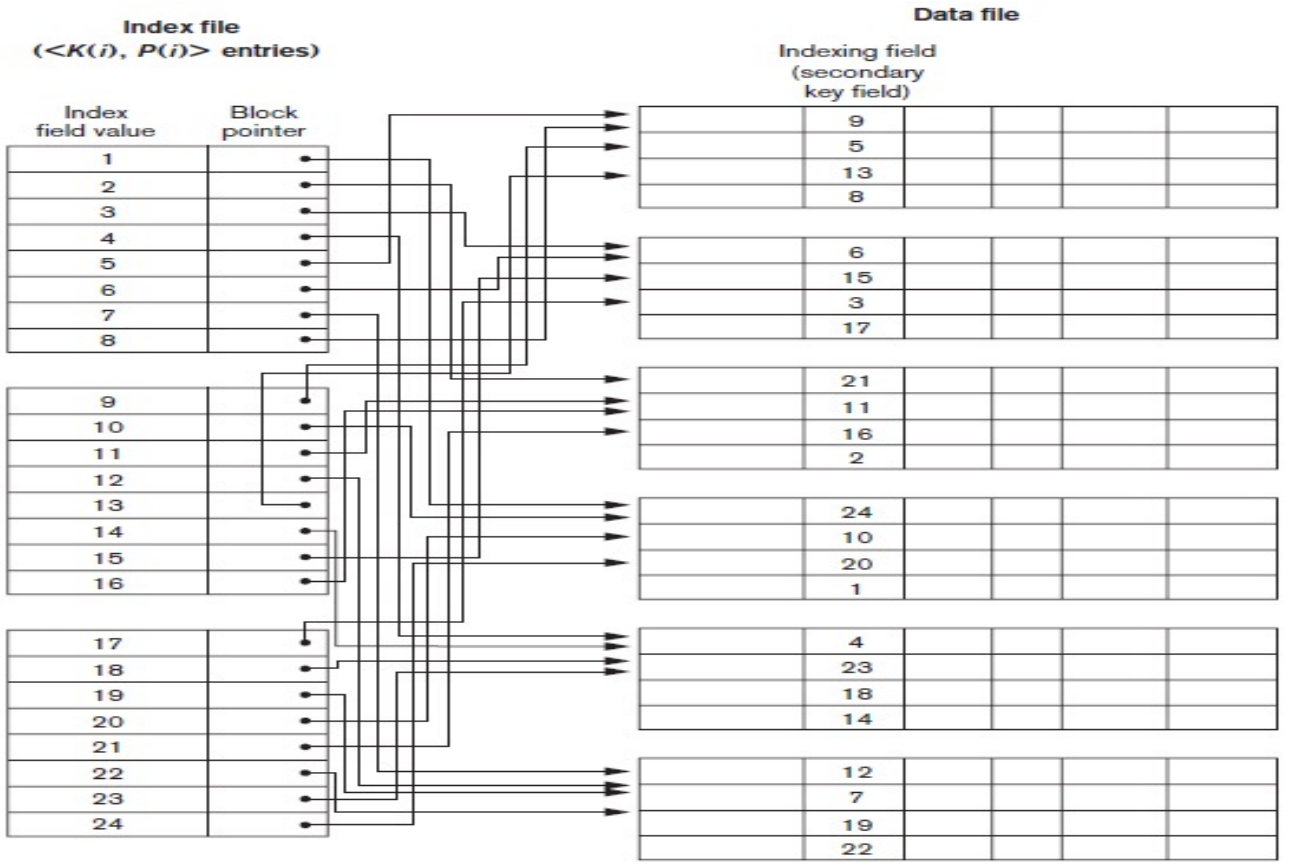
A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file



Indexing in database systems

- **Secondary Index**: generated from a field which is a candidate key or a non-key

Dense secondary index (with block pointers) on a nonordering key field of a file



Indexing in database systems: Types of single-level ordered indexes

	Index Field Used for Physical Ordering of the File	Index Field Not Used for Physical Ordering of the File
Indexing field is key	Primary index	Secondary index (Key)
Indexing field is nonkey	Clustering index	Secondary index (NonKey)

Types of indexes based on the properties of the indexing field

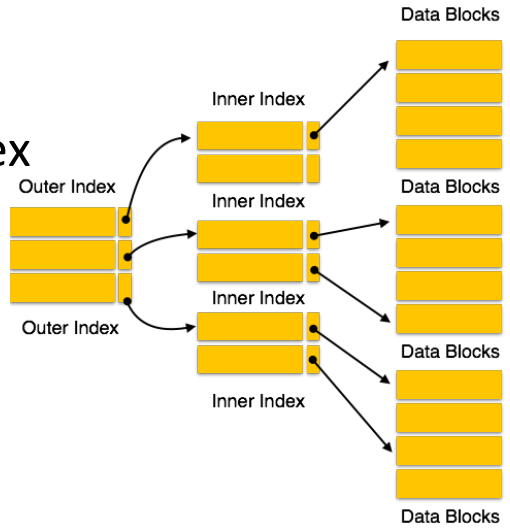
Type of Index	Number of (First-Level) Index Entries	Dense or Nondense (Sparse)	Block Anchoring on the Data File
Primary	Number of blocks in data file	Nondense	Yes
Clustering	Number of distinct index field values	Nondense	Yes/no[a]
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	Number of records[b] or number of distinct index field values[c]	Dense or Nondense	No

[a]Yes if every distinct value of the ordering field starts a new block; no otherwise.
[b]For option 1.
[c]For options 2 and 3.

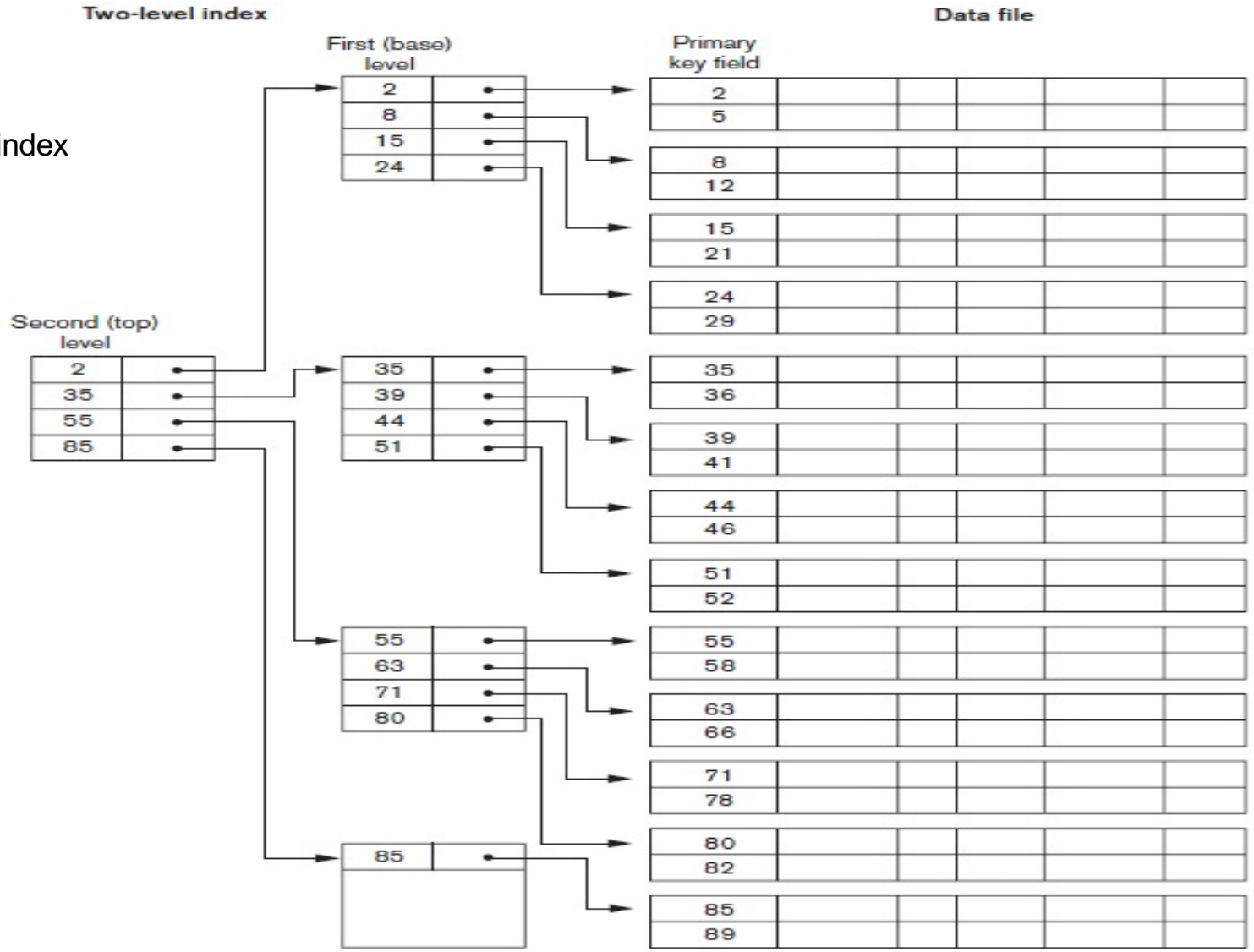
Properties of index types

Indexing in database systems: Multilevel indexes

- Designed to greatly reduce remaining search space as search is conducted
- Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory
  - Index file: Considered first (or base level) of a multilevel index
  - Second level: Primary index to the first level
  - Third level: Primary index to the second level

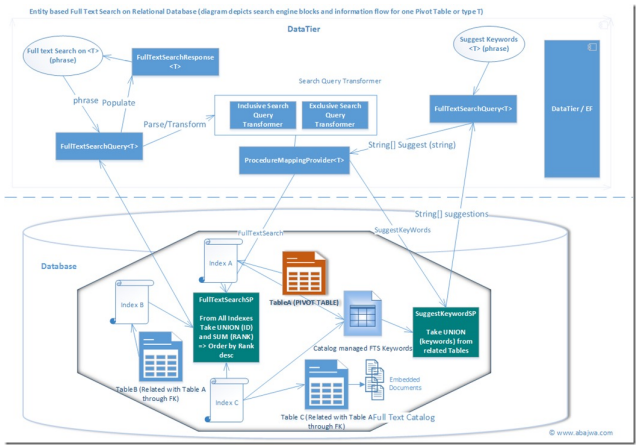


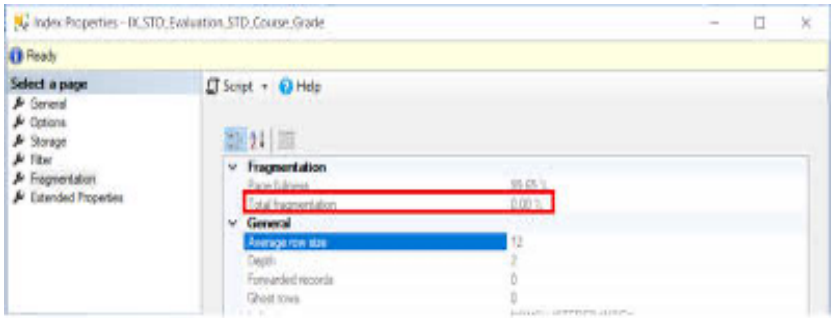
A two-level primary index organization

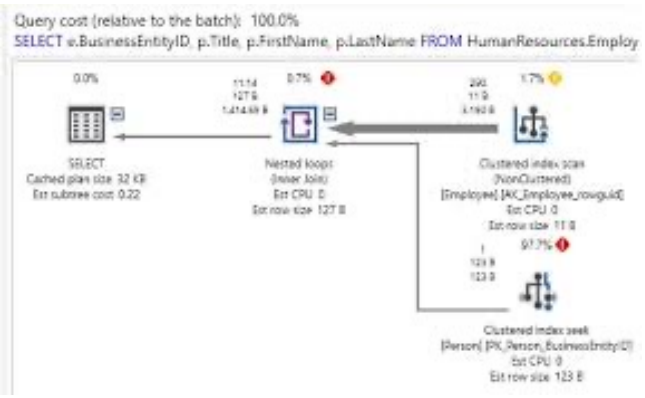


Indexing in database systems: Evaluation

- For all operations (search/select, delete, modify/update):
  - CPU-cost
  - I/O cost
  - Memory cost
  - Datasets: uniform or skew distribution?, etc.
- "Representative" cost(s) can be used (e.g., number of data objects obtained)
- Quite relevant to TPC synthetic benchmarks: a great performance evaluation benchmark → measure the throughput and response times of DBMS







Indexing in database systems: General issues

- Physical index
  - Pointer specifies physical record address
  - Disadvantage: pointer must be changed if record is moved
- Logical index
  - Used when physical record addresses expected to change frequently
  - Entries of the form (K, K_p)
- Physical DB/index design very much depends on the system
- Concurency control & security/privacy related issues
  - Outsourced DB services

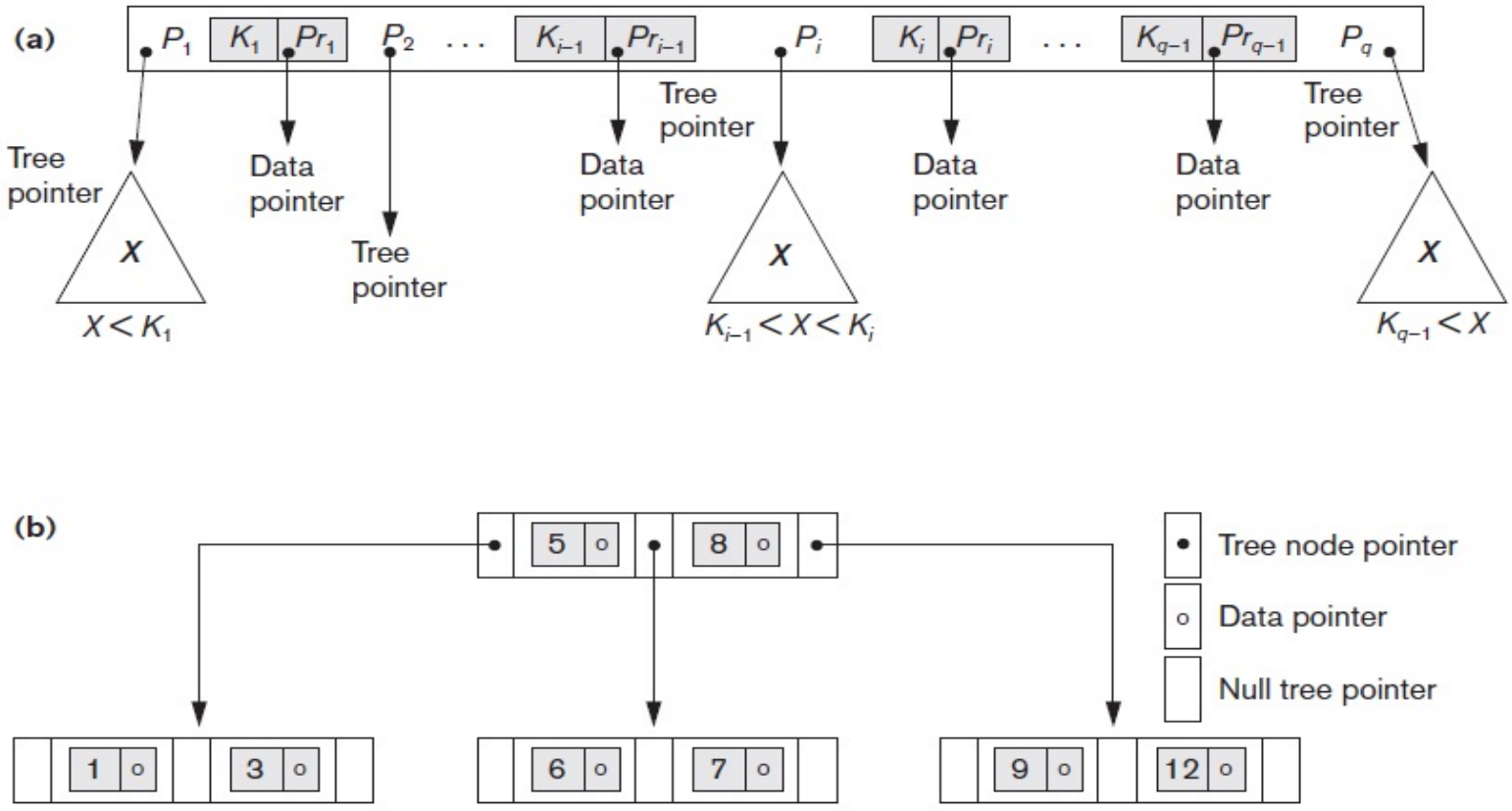
Outline

- ~~Indexing in database systems~~
- Traditional indexing: B/B^+ tree
- Spatial indexing: R/R^* tree
- Indexing techniques for high-dimensional data
  - Curse of dimensionality
- New indexing techniques for emerging application domains
- Exercises
- Reading: chapters 16, 17 [1] and internet

B/B$^+$ tree

- Dynamic multilevel indexes
- Tree data structure terminology
  - Tree is formed of nodes
  - Each node (except root) has one parent and zero or more child nodes
  - Leaf node has no child nodes
    - Unbalanced if leaf nodes occur at different levels
  - Nonleaf node called internal node
  - Subtree of node consists of node and all descendant nodes

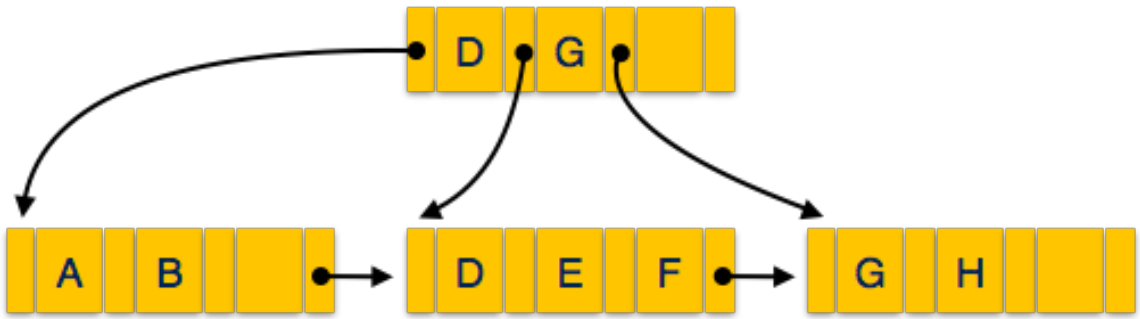
B-tree



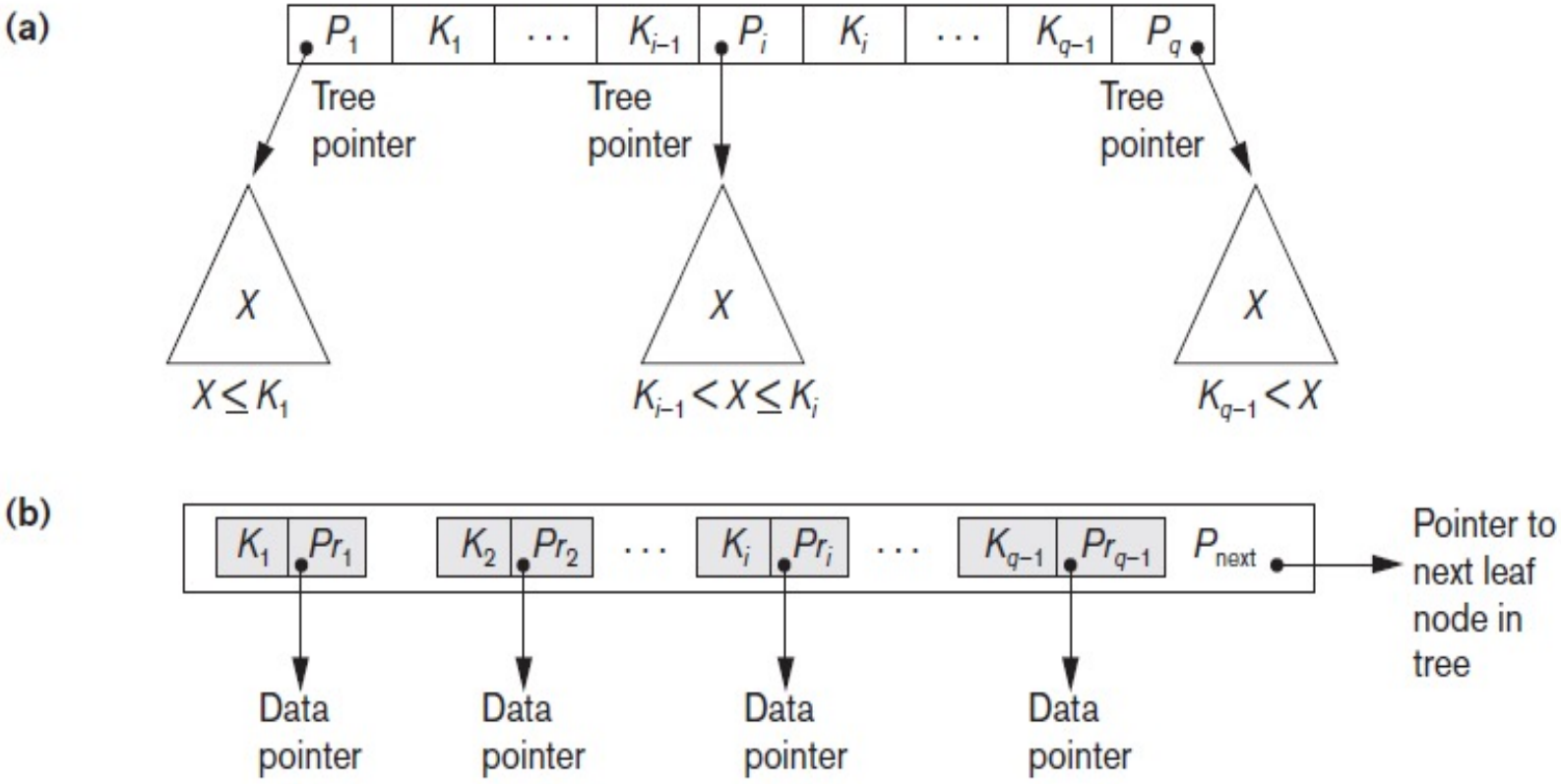
B-tree structures (a) A node in a B-tree with *q−1* search values (b) A B-tree of order *p=3.* The values were inserted in the order 8, 5, 1, 7, 3, 12, 9, 6

B^+ tree

- B^+ tree is a balanced binary search tree that follows a multi-level index format
- Leaf nodes consist of actual data pointers and all leaf nodes remain at the same height, thus balanced
- The leaf nodes are linked using a link list

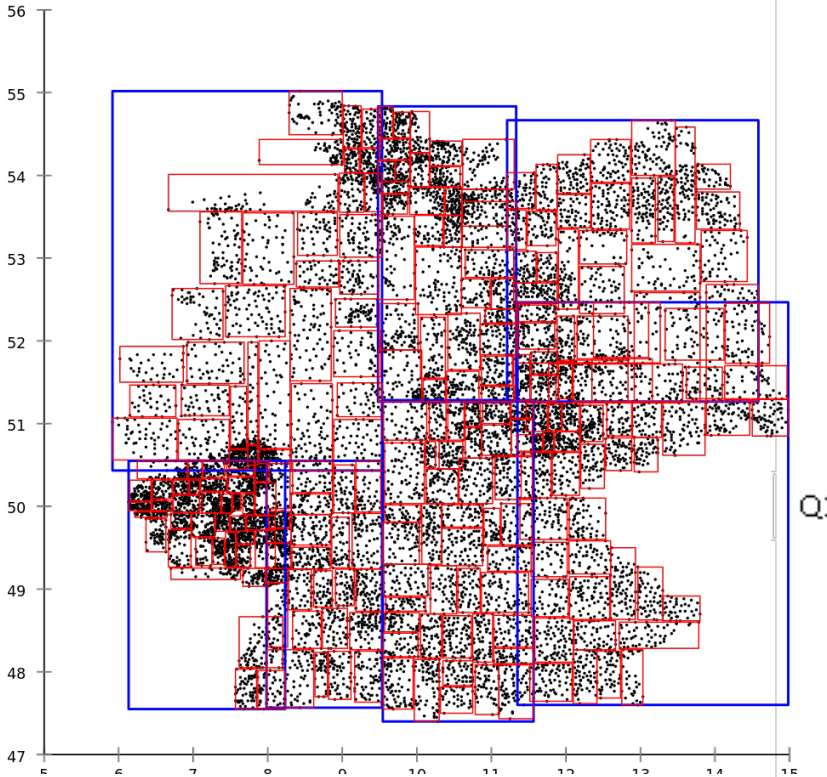


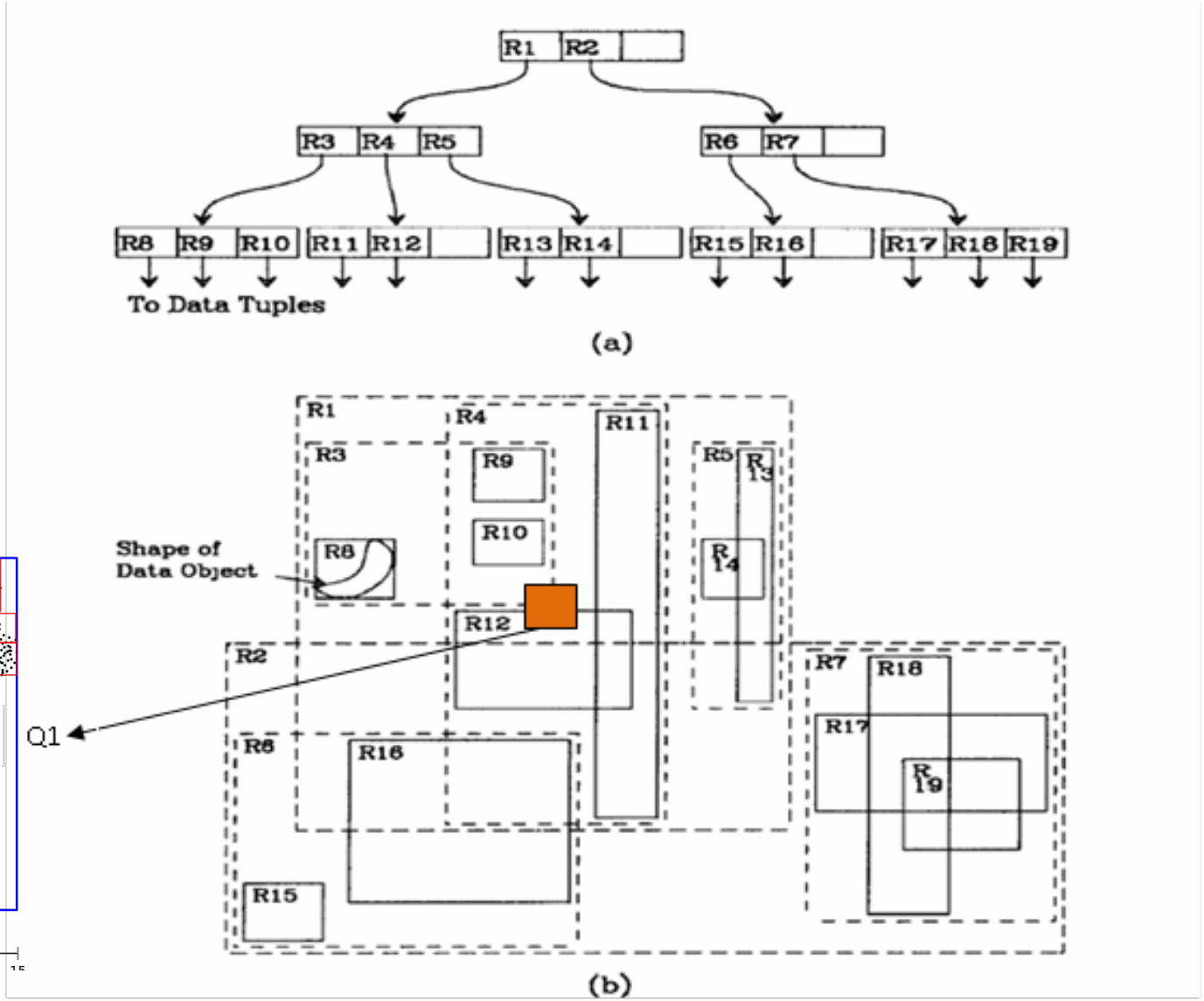
B+-tree



The nodes of a B+-tree (a) Internal node of a B+-tree with *q−1* search values (b) Leaf node of a B+-tree with *q−1* search values and *q−1* data pointers

R/R* tree


56
55
54
53
52
51
50
49
48
47


R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
To Data Tuples
(a)
Shape of
Data Object
Q1
(b)

Outline

- ~~Indexing in database systems~~
- ~~Traditional indexing: B/B$^+$ tree~~
- ~~Spatial indexing: R/R* tree~~
- Indexing techniques for high-dimensional data
  - Curse of dimensionality
- New indexing techniques for emerging application domains
- Exercises
- Reading: chapters 16, 17 [1] and internet

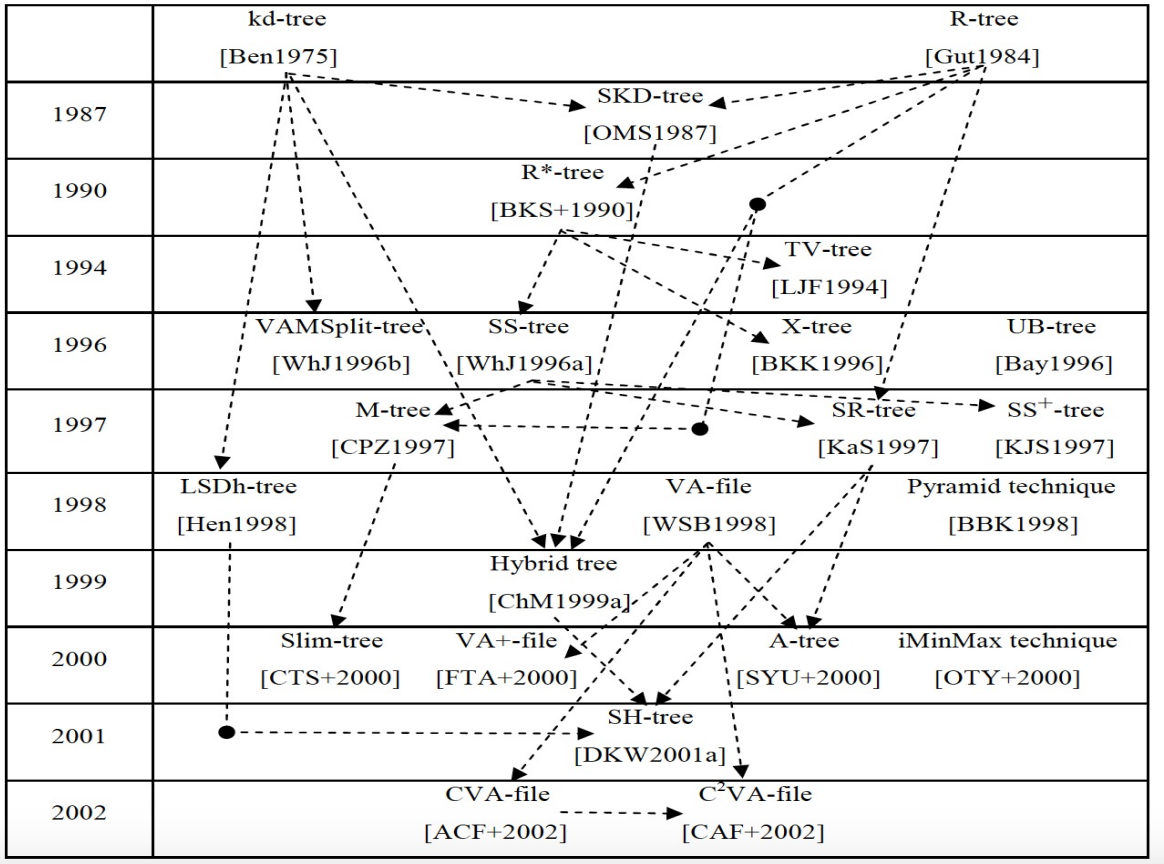
Indexing techniques for high-dimensional data - Curse of dimensionality

- A measure (e.g., Euclidean distance) is defined using many coordinates, there is little difference in the distances between different pairs of data objects
- The distance between the center and the corners of a d-dimensional hepercube with edges of length 2r is $r\sqrt{d}$
  - increases without bound for fixed r
  - nearly all of the high-dimensional space is "far away" from the centre
  - the high-dimensional unit hypercube can be said to consist almost entirely of the "corners" of the hypercube

→a “query” will overlap with almost entire space !

→Solution: new indexing techniques, dimensionality reduction, etc.

Indexing techniques for high-dimensional data



- **DANG Tran Khanh**: *Semantic Based Similarity Searches in Database Systems (Multidimensional Access Methods, Similarity Search Algorithms)*. PhD thesis, FAW-Institute, Johannes Kepler University of Linz, Austria, May 2003
- Volker Gaede, Oliver Günther: *Multidimensional access methods*, ACM Computing Surveys (CSUR), 30(2), June 1998

Outline

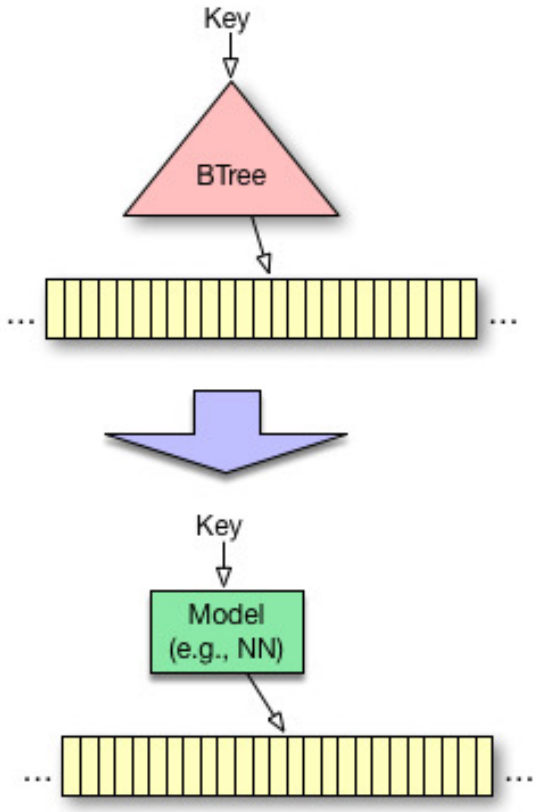
- ~~Indexing in database systems~~
- ~~Traditional indexing: B/B$^+$ tree~~
- ~~Spatial indexing: R/R* tree~~
- ~~Indexing techniques for high-dimensional data~~
  - ~~Curse of dimensionality~~
- New indexing techniques for emerging application domains
- Exercises
- Reading: chapters 16, 17 [1] and internet

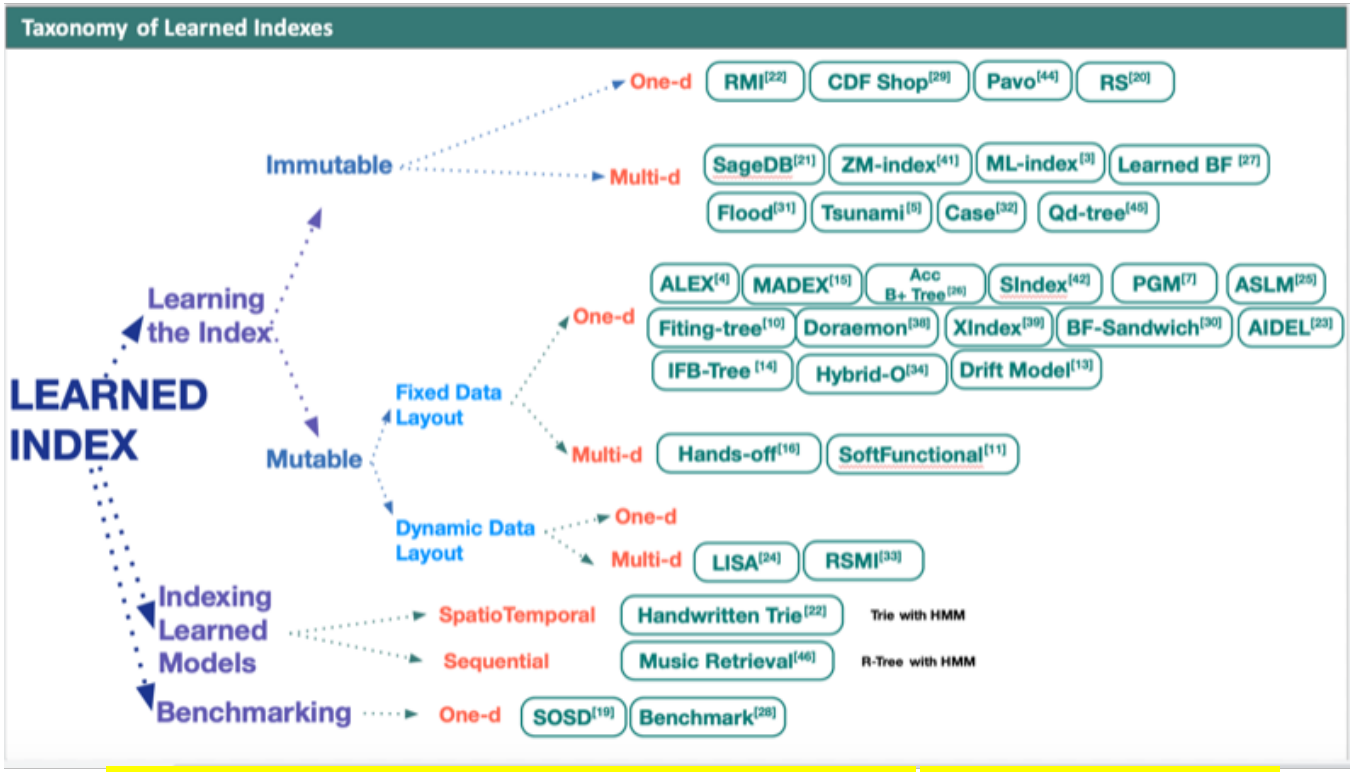
New indexing techniques for emerging application domains

- Big data & apps
  - Open data
  - Spatio-temporal data
  - IoT & smart cities
  - Distributed ledgers such as blockchain-based data
- Further reading: Learned index structures
  - Idea: https://research.google/pubs/pub46518/
  - Benchmark: https://vldb.org/pvldb/vol14/p1-marcus.pdf

New indexing techniques for emerging application domains

- Learned index structures





https://www.cs.purdue.edu/homes/aref/learned-indexes-tutorial.html

Exercises

- Given in the class, group working