RDBMS and SQL

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Relational Algebra

SQL

Designing tables

Lecture 9, 28 October 2026

Storing data

- RAM vs Hard disk vs SSD
- Blocks and latency

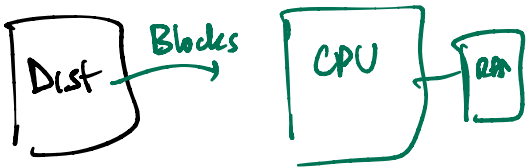
RAM ~ speed of CPU

10^9 data items / sec

$\simeq 10^3$ ~ 4096 B

Hard Disk ~ 10^3

SSD ~ 10^6



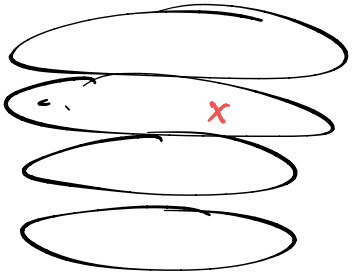
Storing data

- RAM vs Hard disk vs SSD
- Blocks and latency

Disk challenge

Find the place on disk to read SEEK

Then transfer data READ / WRITE



Storing data

- RAM vs Hard disk vs SSD
- Blocks and latency

Aadhaar 10^9 rows $\Rightarrow$ 1000 rows/block

10^6 blocks $\sim 10^3$ secs

Storing data

- RAM vs Hard disk vs SSD
- Blocks and latency

Complexity of DB algorithms

Only measure I/O cost

"All ops in memory are free"

Count "seeks" & "reads"

Fixed length records = rows

- Blocks and block boundaries

Don't split rows across blocks

N rows

B block size

R row length

B/R rows per block

Need N/(B/R) blocks

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

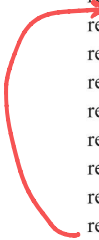
Deleting a record

- Remove Einstein

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	~~22222~~	~~Einstein~~	~~Physics~~	~~95000~~
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Deleting a record

- Remove Einstein
- Compress



record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

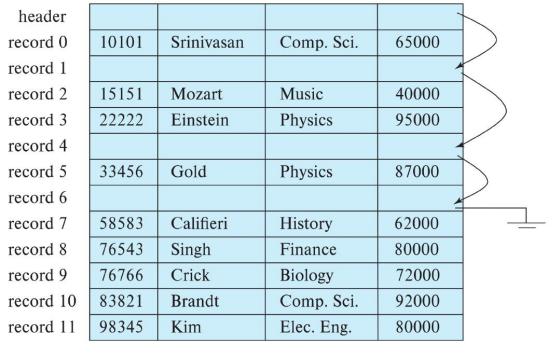
Deleting a record

- Remove Einstein
- Compress
- Move last record

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

Deleting a record

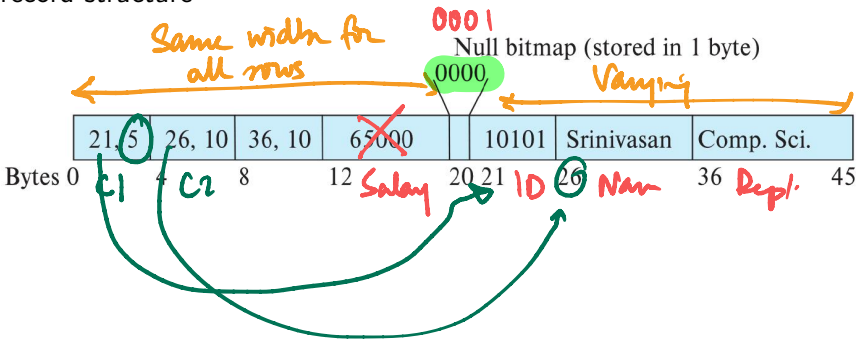
- Remove Einstein
- Compress
- Move last record
- Maintain free list of empty slots



header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

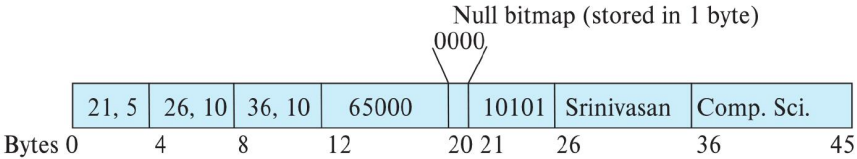
Variable length records

- Single record structure

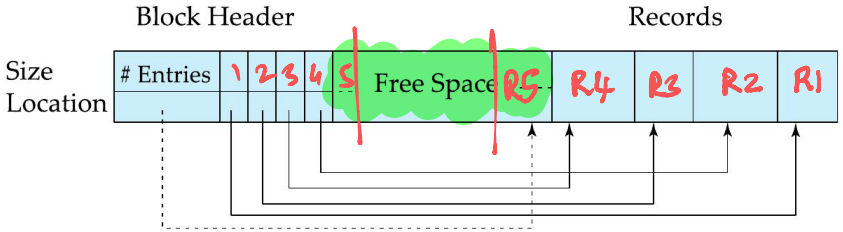


Variable length records

- Single record structure



- Slotted page block structure



Storing tables — heap file organization

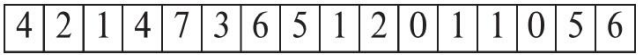
- Rows are distributed across blocks in no fixed order

Storing tables — heap file organization

- Rows are distributed across blocks in no fixed order
- Use first available free slot

Storing tables — heap file organization

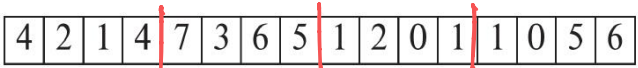
- Rows are distributed across blocks in no fixed order
- Use first available free slot
- Maintain map of free space per block
  - Interpret as fraction that is free — using 3 bits, 7 means 7/8 free



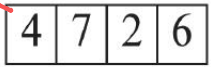
3 bit map
0-7

Storing tables — heap file organization

- Rows are distributed across blocks in no fixed order
- Use first available free slot
- Maintain map of free space per block
  - Interpret as fraction that is free — using 3 bits, 7 means 7/8 free



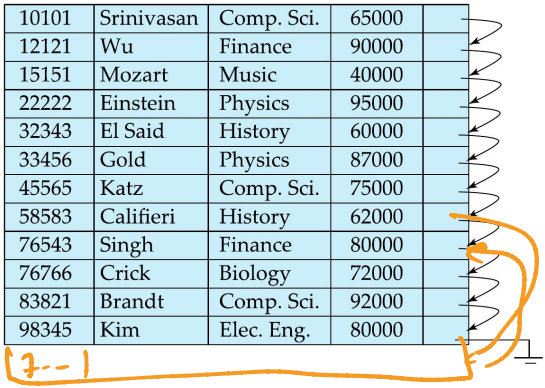
- Second level free space map



max per segment

Storing tables — sequential file organization

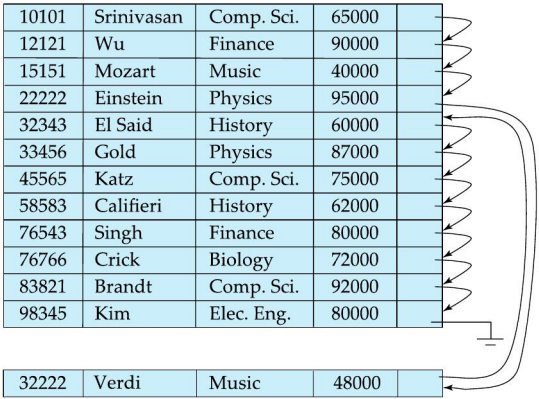
- Rows are ordered sequentially by key
  - Each row points to next key value
  - Allows some flexibility



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Storing tables — sequential file organization

- Rows are ordered sequentially by key
  - Each row points to next key value
  - Allows some flexibility
- Overflow block

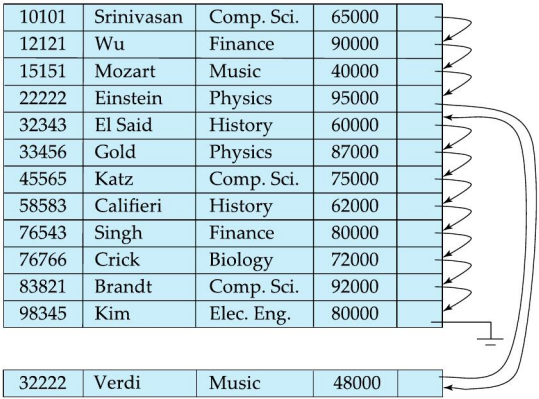


10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

32222	Verdi	Music	48000	

Storing tables — sequential file organization

- Rows are ordered sequentially by key
  - Each row points to next key value
  - Allows some flexibility
- Overflow block
- Periodically, reorganize the entire table



Data dictionary / system catalogue

Need to store metadata

- Table names
- Attribute details — names, type, size
- Views
- Integrity constraints
- Descriptive statistics about tables — number of tuples, distinct values, ...
- Information about indices

Indexing

- Why build an index?

Indexing

- Why build an index?
- Search key
  - As opposed to superkey, candidate key, ...
  - May need multiple search keys for a table

Indexing

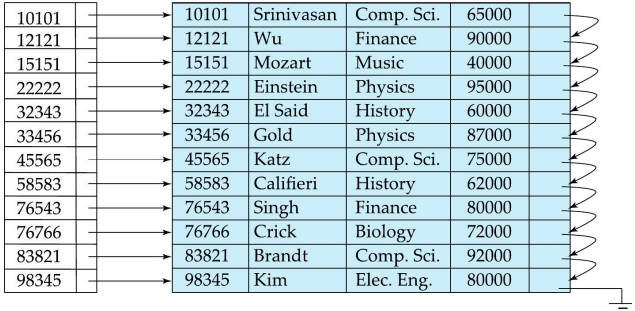
- Why build an index?
- Search key
  - As opposed to superkey, candidate key, ...
  - May need multiple search keys for a table
- Types of queries — point vs range
  - `ID = "10102"`
  - `salary > 75000`

Indexing

- Why build an index?
- Search key
  - As opposed to superkey, candidate key, ...
  - May need multiple search keys for a table
- Types of queries — point vs range
  - `ID = "10102"`
  - `salary > 75000`
- Maintaining an index
  - Inserts, deletes
  - Space

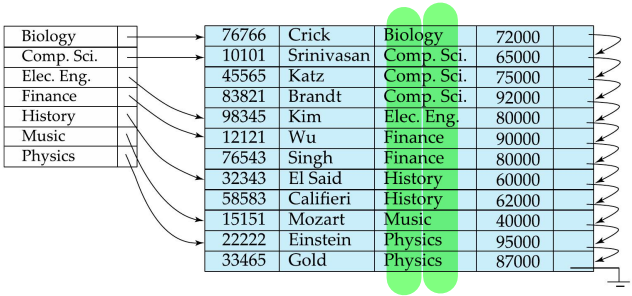
Clustering index

- File is ordered with respect to index values
- Index sequential file
- Dense index — every value is present in the index



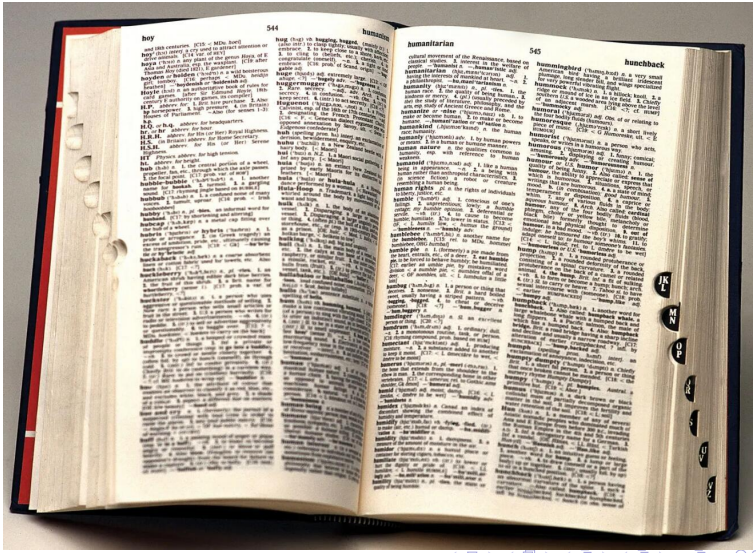
Clustering index

- File is ordered with respect to index values
- Index sequential file
- Dense index — every value is present in the index
  - Index value may match multiple records



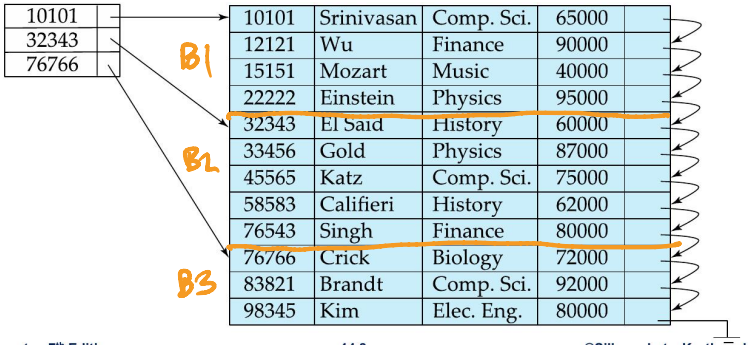
Indexing — sparse indices

- Maintain indices for a subset of values
  - Page headers in a dictionary



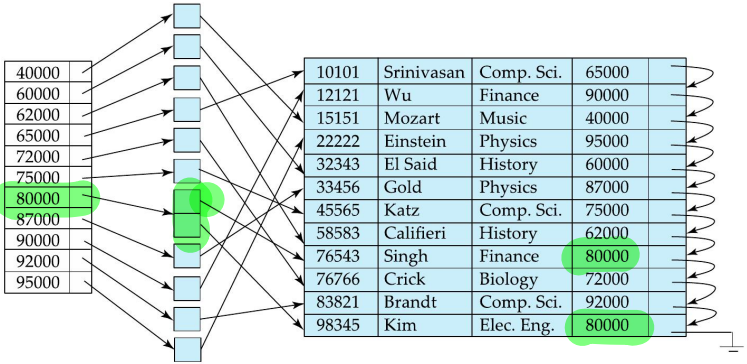
Indexing — sparse indices

- Maintain indices for a subset of values
  - Page headers in a dictionary
- Align to block boundaries
  - Records are still sequential with respect to index
  - Sparse index identifies first record in each block



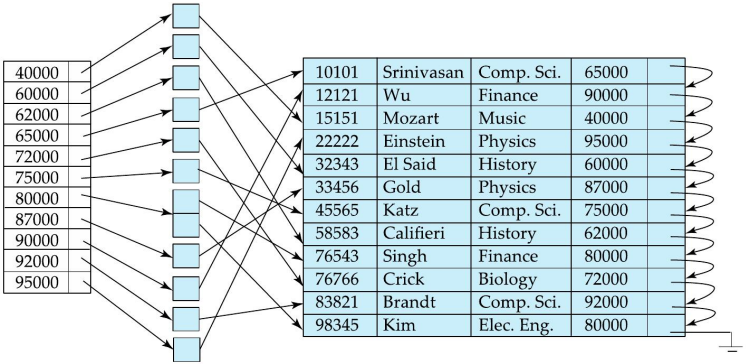
Indexing — secondary index

- Index for an attribute that does not match sequence in which table is stored



Indexing — secondary index

- Index for an attribute that does not match sequence in which table is stored
- Key points to block that contains pointers to matching records
  - Can have multiple records for same search key

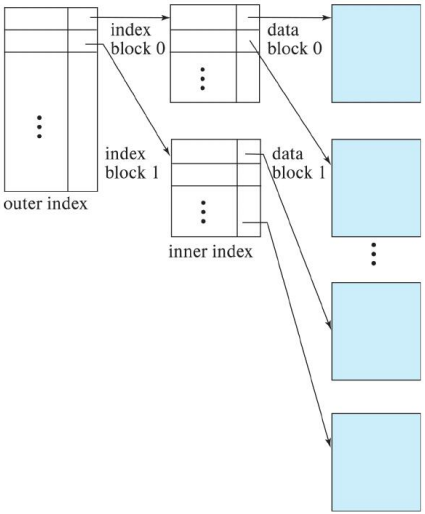


Storage

- Typically, index will not fit in RAM

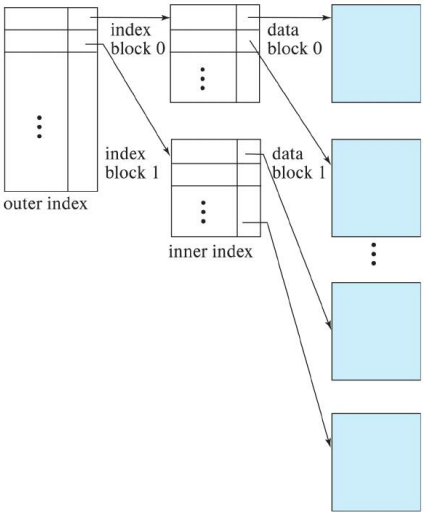
Storage

- Typically, index will not fit in RAM
- Store index as a sequential file
  - Build a sparse index for the index file
  - Multi-level, till sparse index fits in one block



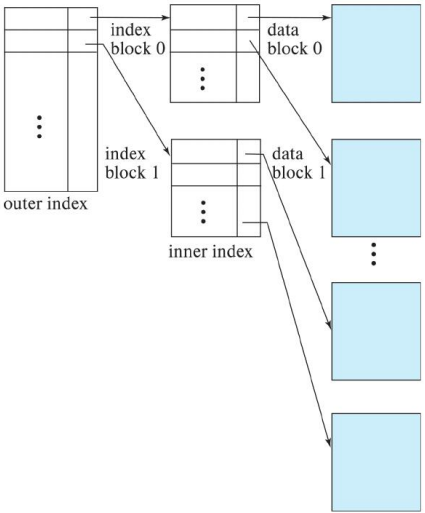
Storage

- Typically, index will not fit in RAM
- Store index as a sequential file
  - Build a sparse index for the index file
  - Multi-level, till sparse index fits in one block
- Binary search to find required key



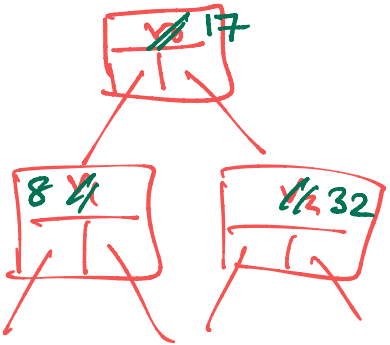
Storage

- Typically, index will not fit in RAM
- Store index as a sequential file
  - Build a sparse index for the index file
  - Multi-level, till sparse index fits in one block
- Binary search to find required key
- Idea leads to a more efficient structure



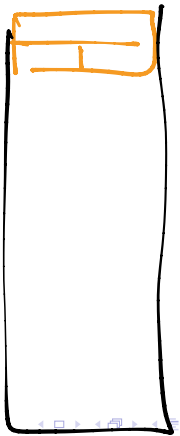
Search trees

- Binary search trees
  - Binary search on dynamic data
  - Balanced tree has logarithmic height



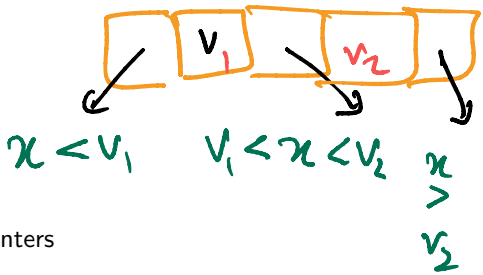
Search trees

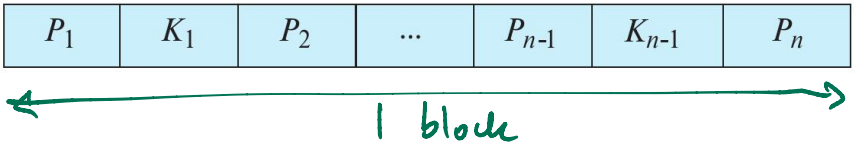
- Binary search trees
  - Binary search on dynamic data
  - Balanced tree has logarithmic height
- Block-based access
  - Binary tree node has one search key value, two pointers
  - Block can hold much more



Search trees

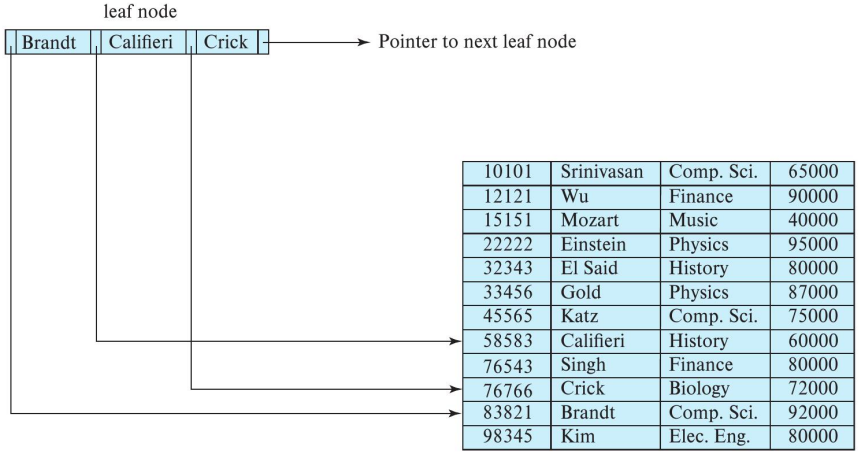
- Binary search trees
  - Binary search on dynamic data
  - Balanced tree has logarithmic height
- Block-based access
  - Binary tree node has one search key value, two pointers
  - Block can hold much more
- Generalize to multiple key values, multiple pointers





B+ trees

- Leaf nodes form a dense index — linked list of leaves, each one block

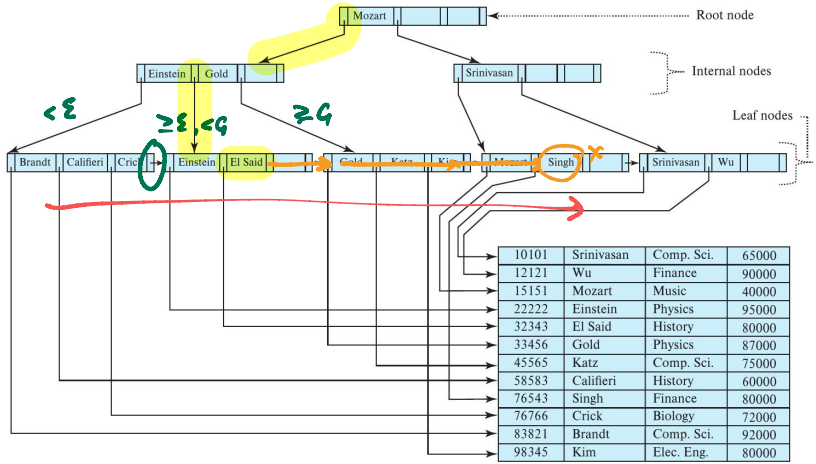


instructor file

B+ trees

- Leaf nodes form a dense index — linked list of leaves
- Non-Leaf nodes form a sparse index

names between El Said & Singh

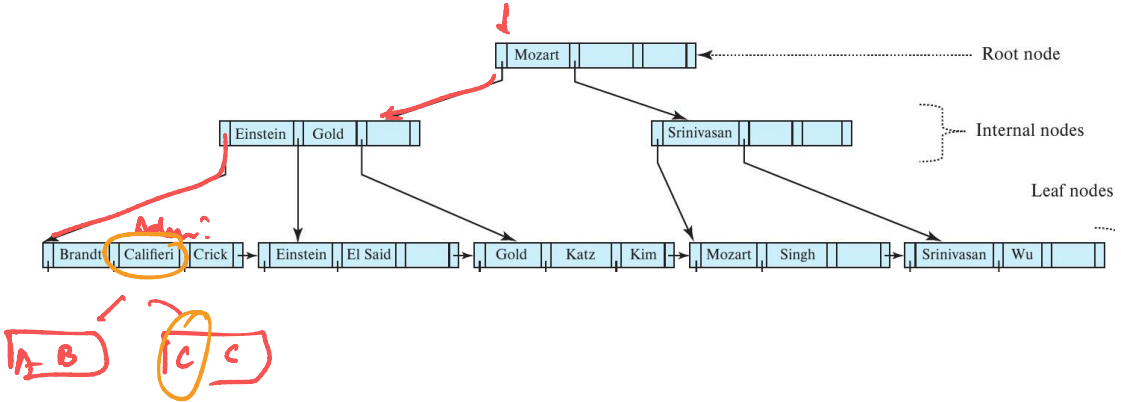


B+ trees

- Leaf nodes form a dense index — linked list of leaves
- Non-leaf nodes form a sparse index
- Constraints — assume n keys and pointers can fit in a block
  - Each leaf has at least $\lceil (n-1)/2 \rceil$ key values
  - Each non-leaf has at least $\lceil n/2 \rceil$ pointers
  - Height of the tree is proportional to $\log_{n/2}(n)$

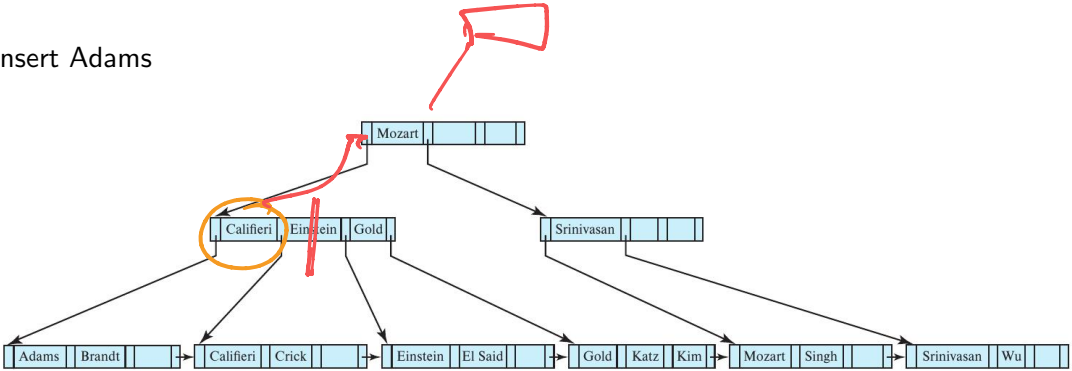
B+ trees — insertion

- Insert Adams



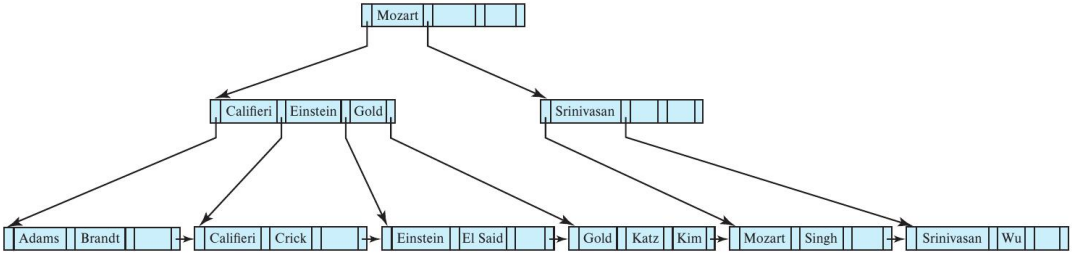
B+ trees — insertion

- Insert Adams



B+ trees — insertion

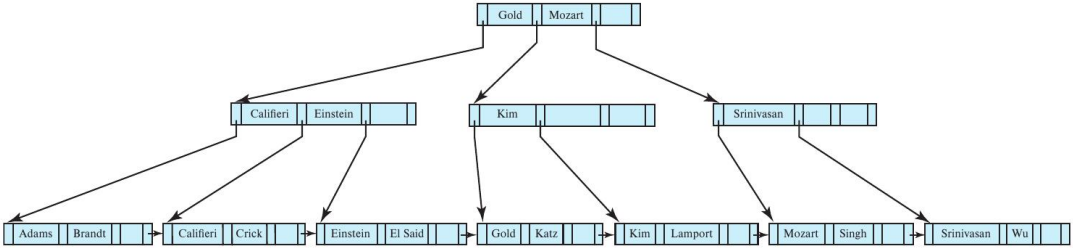
- Insert Adams



- Insert Lamport

B+ trees — insertion

- Insert Adams
- Insert Lamport

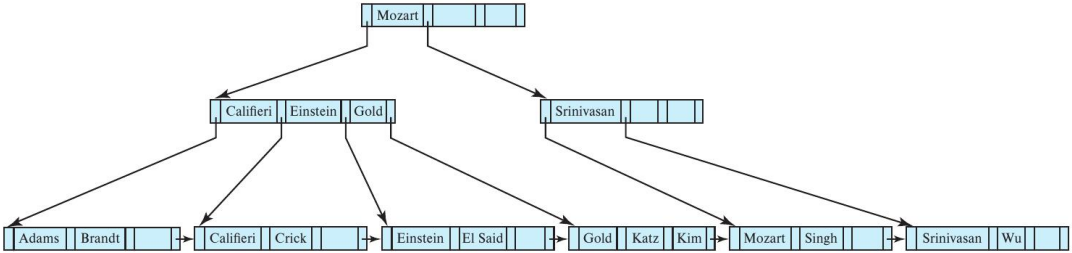


B+ trees — insertion

- Insert Adams
- Insert Lamport
- Recursively insert from leaf level upwards
  - Split nodes when needed and adjust search keys and pointers

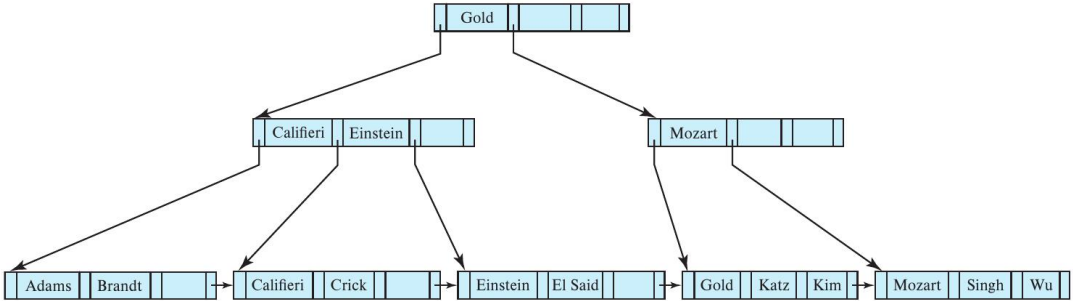
B+ trees — deletion

- Delete Srinivasn



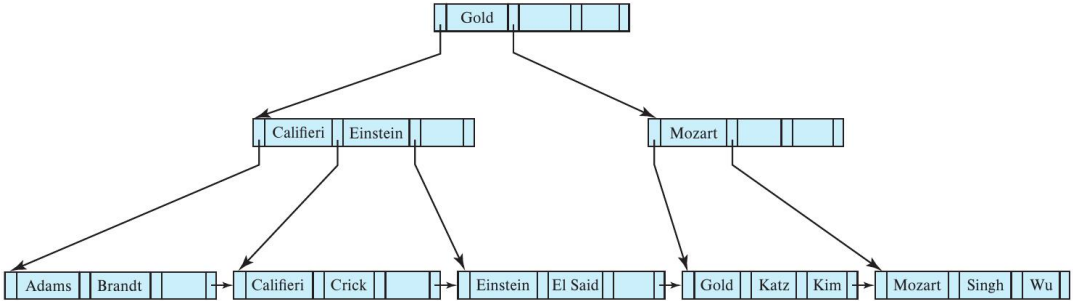
B+ trees — deletion

- Delete Srinivasn



B+ trees — deletion

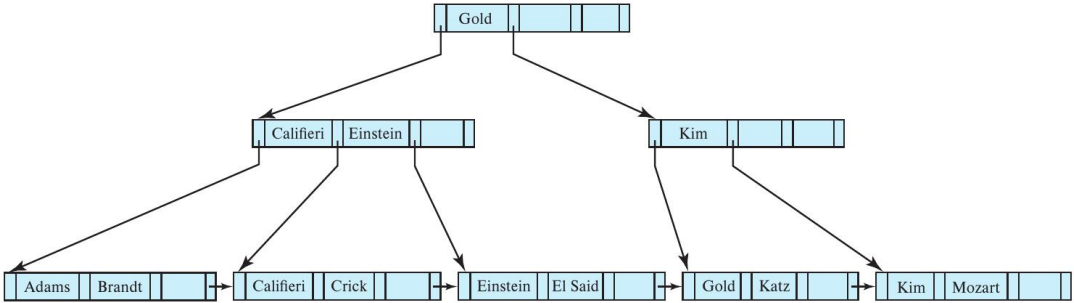
- Delete Srinivasn



- Delete Singh and Wu

B+ trees — deletion

- Delete Srinivasn
- Delete Singh and Wu

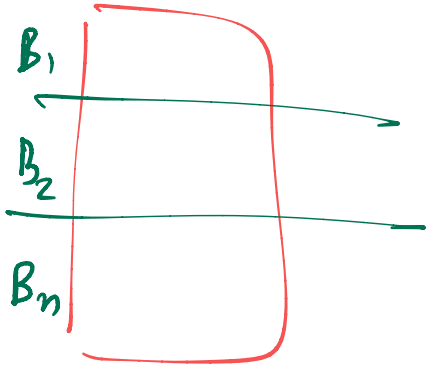


B+ trees — deletion

- Delete Srinivasn
- Delete Singh and Wu
- Recursively delete from leaf level upwards
  - Merge or redistribute with neighbour

Hashing

$h(k_i) = B_2$



Creating an index in SQL

- Not part of SQL standard

Creating an index in SQL

- Not part of SQL standard
- `create index I on T(A,B,C)`

Creating an index in SQL

- Not part of SQL standard
- `create index I on T(A,B,C)`
- `drop index I`

Creating an index in SQL

- Not part of SQL standard
- `create index I on T(A,B,C)`
- `drop index I`
- DBMS may create and maintain index on its own for efficient processing