Question 3c

- `Product(maker, model, price)`
- `PC(model, speed)`
- `Printer(model, type)`

Write in SQL: For each maker, find the minimum and maximum price of a (PC, ink-jet printer) combination.

```
SELECT p1.maker, min(p1.price+p2.price), max(p1.price+p2.price)
FROM  Product p1, Product p2, PC pc, Printer t
WHERE  t.type = 'ink-jet' and p1.model = pc.model and p2.model =
          t.model and p1.maker=p2.maker
GROUP BY  p1.maker
```

2

Question 4

Given the instance of two relations:

R:

A	B
1	2
3	4
1	3

S:

B	C
1	3
2	4

a) What is the result of the following query:

```
SELECT DISTINCT R.A
FROM R
WHERE R.A NOT IN (SELECT DISTINCT S.B AS A
                  FROM S
                  WHERE S.B = S.C)
```

A
1
3

3

Question 4b

R:

A	B
1	2
3	4
1	3

S:

B	C
1	3
2	4

```
SELECT R.A, S.C, avg(R.B) as av
FROM R, S
WHERE R.B < 4
GROUP BY R.A, S.C
HAVING max(R.B) >= 2
```

A	R.B	S.B	C
1	2	1	3
1	2	2	4
~~3~~	~~4~~	~~1~~	~~3~~
~~3~~	~~4~~	~~2~~	~~4~~
1	3	1	3
1	3	2	4

A	R.B	S.B	C
1	2	1	3
1	3	1	3
1	2	2	4
1	3	2	4

A	C	av
1	3	2.5
1	4	2.5

4

Question 5

Consider the following CREATE TABLE definition:

```
CREATE TABLE Midterm
  (A INT NOT NULL,
   B INT NOT NULL,
   C INT NOT NULL,
   PRIMARY KEY (A),
   FOREIGN KEY (B) REFERENCES Midterm(A) ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (C) REFERENCES Midterm(A) ON DELETE CASCADE ON UPDATE RESTRICT)
```

Consider the following instance table Midterm:

A	B	C
4	3	3
3	4	3

a) What is the result of the following statement:

```
UPDATE Midterm
SET B = B+1
WHERE B in (SELECT A FROM Midterm)
```

Answer: Error, the FK constraint is violated

5

6

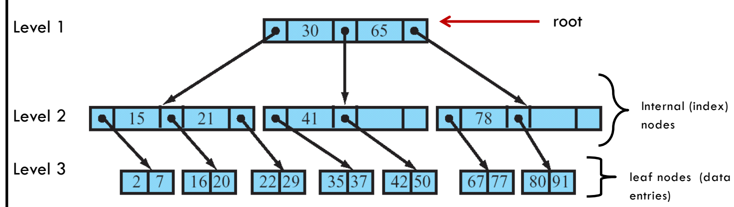
End

6

B+ Tree Index

7

- The B+ tree structure is the most common index type in databases
- Index files can be quite large, often stored on disk, partially loaded into memory as needed
- Each node is at least 50% full



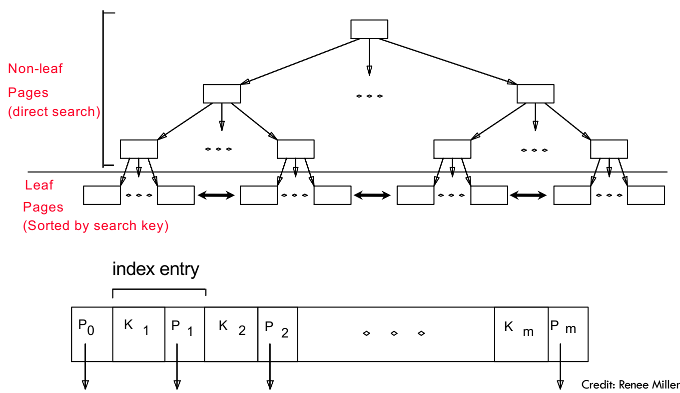
Credit: S. Lee

7

B+ Tree Index

8

Supports equality and range-searches efficiently

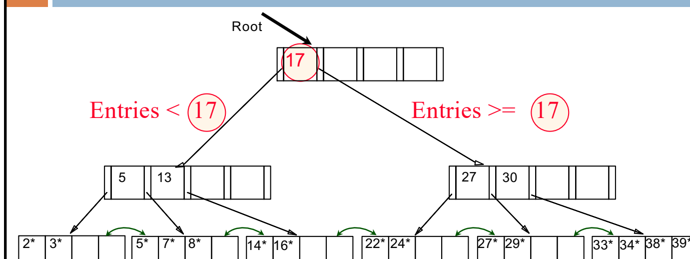


Credit: Renee Miller

8

Example

9

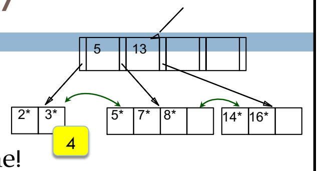


- Find 28*? 29*? All > 15* and < 30*
- Insert/delete: Find data entry in leaf, then change it. Need to adjust parent sometimes.
  - And change sometimes bubbles up the tree

9

Inserting a Data Entry

10



- Find correct leaf L.
- Put data entry onto L.
  - If L has enough space, done!
  - Else, must *split* L (into L and a new node L2)
    - Redistribute entries evenly, copy up middle key.
    - Insert index entry pointing to L2 into parent of L.
- This can happen recursively
  - To split index node, redistribute entries evenly, but push up middle key.
- Splits "grow" tree; root split increases height.

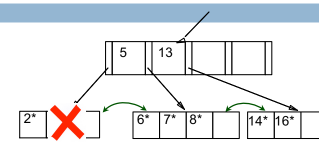
Insert data value 4

10

Deleting a Data Entry

Delete value 3

11



- Start at root, find leaf L where entry belongs.
- Remove the entry.
  - If L is at least half-full, done!
  - If not,
    - Try to re-distribute, borrowing from sibling (adjacent node with same parent as L).
    - If re-distribution fails, merge L and sibling.
- If merge occurred, must delete entry (pointing to L or sibling) from parent of L.
- Merge could propagate to root, decreasing height.

11

Balanced vs. Unbalanced Trees

12

- In a balanced tree, every path from the root to a leaf node is the same length.

```
o Balanced                              o Unbalanced
                                                          [===]
          [===]                                          /
         /     \                                    [===]
     [===]     [===]                               /     \
    /     \    /   \                          [===]     [===]
[===]  [===] [===]  [===]                    /
                                        [===]
                                       /     \
                                   [===]     [===]
```

Credit: S. Lee

12

Hash Based Indexes

13

- Good for equality selections.
- Index is a collection of buckets
  - Bucket = primary page plus zero or more overflow pages.
  - Buckets contain data entries.
- Hashing function h: h(r) = bucket in which (data entry for) record r belongs. h looks at the search key fields of r.
  - No need for "index entries" in this scheme.

13