# Quiz

Hashing , B+trees, and Extendible Hashing

## Advanced Database

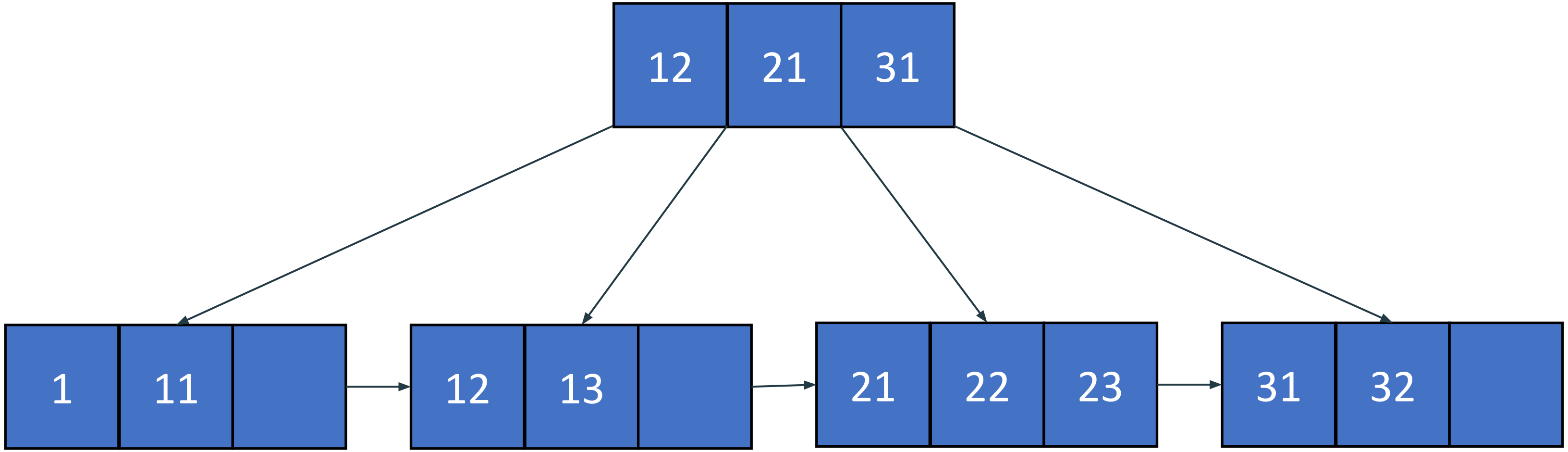
Consider an extendible hashing structure such that:

- Each bucket can hold up to two records
- The hashing function uses the **lowest g bits** (i.e **least significant bits** of the number), where g is the global depth

a. Starting from an empty table, insert 6,15, 34, 18
   i. What is the total depth of the resulting table?
   ii. What is the local depth of the bucket containing 34?
b. Starting from the result in (a), you insert keys 16, 7, 10, 20, 9
   i. Which key will first cause a split without doubling the size of the table?
   ii. Which key will first make the table double the size?

a. Insert 24 to the following B+tree

b. Delete 23 from the B+tree obtained in part a.



- Consider the relations **R(<u>A</u>, B)**, **S(<u>C</u>, D)** and **T(<u>D</u>, B, E)** with keys underlined, i.e., attribute A is a key for R, C is a key for S, and D is a key for T.
- **Tuples from S and T take up 50 bytes**, while **tuples from R comprise 40 bytes**. Values for attribute **S.D take up 10 bytes**.
- The **page size is set to 4000 bytes** and there are **8 pages in the buffer.**
- The statistics show that **S contains 36000 tuples**; that **T contains 10000 tuples**; that **R contains 12000 tuples**.
- Furthermore, values in **S.D are uniformly distributed in the range [0,100],** values in **T.E are uniformly distributed in the range [10000 , 30000]**, and **T has 5000 distinct values for attribute B.** In addition, the following **histogram information is available for R.B**:

| range | number of tuples |
|---|---|
| [0,19] | 5000 |
| [20,39] | 200 |
| [40,59] | 800 |
| [60,79] | 2000 |
| [80,100] | 4000 |

- Moreover, relation S has a clustered B+-tree index on attribute D; relation T has unclustered hash indexes on attributes B and E (separately).
- Assume that we only consider **left-deep plans**, and two join algorithms (**block nested-loops**, and **sort-merge join**) only. Use **Selinger Optimizer (dynamic programing algorithm)** to find the optimal cost-based query-plan to the following SQL query (i.e. find the best query tree and the algorithm associated with each operator in the tree).

- **SELECT R.A, S.C, T.D FROM R,S, T**
  **WHERE R.B==30 AND S.D <=60 AND T.E != 10000 AND R.B == T.B AND S.D == T.D**

# Answers

Consider an extendible hashing structure such that:

- Each bucket can hold up to two records
- The hashing function uses the **lowest g bits** (i.e **least significant bits** of the number), where g is the global depth

a. Starting from an empty table, insert 6,15, 34, 18

   i. What is the total depth of the resulting table? ⇒ **3**

   ii. What is the local depth of the bucket containing 34? ⇒ **3**

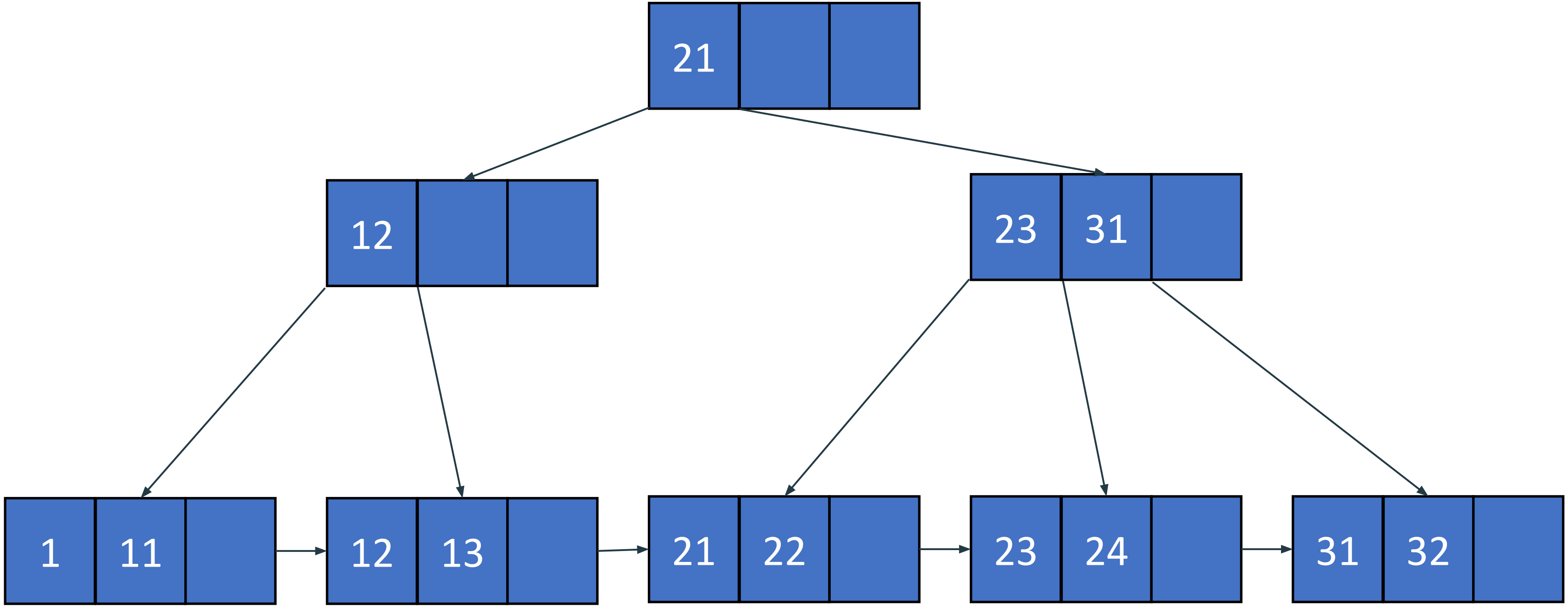
b. Starting from the result in (a), you insert keys 16, 7, 10, 20, 9

   i. Which key will first cause a split without doubling the size of the table? ⇒ **9**

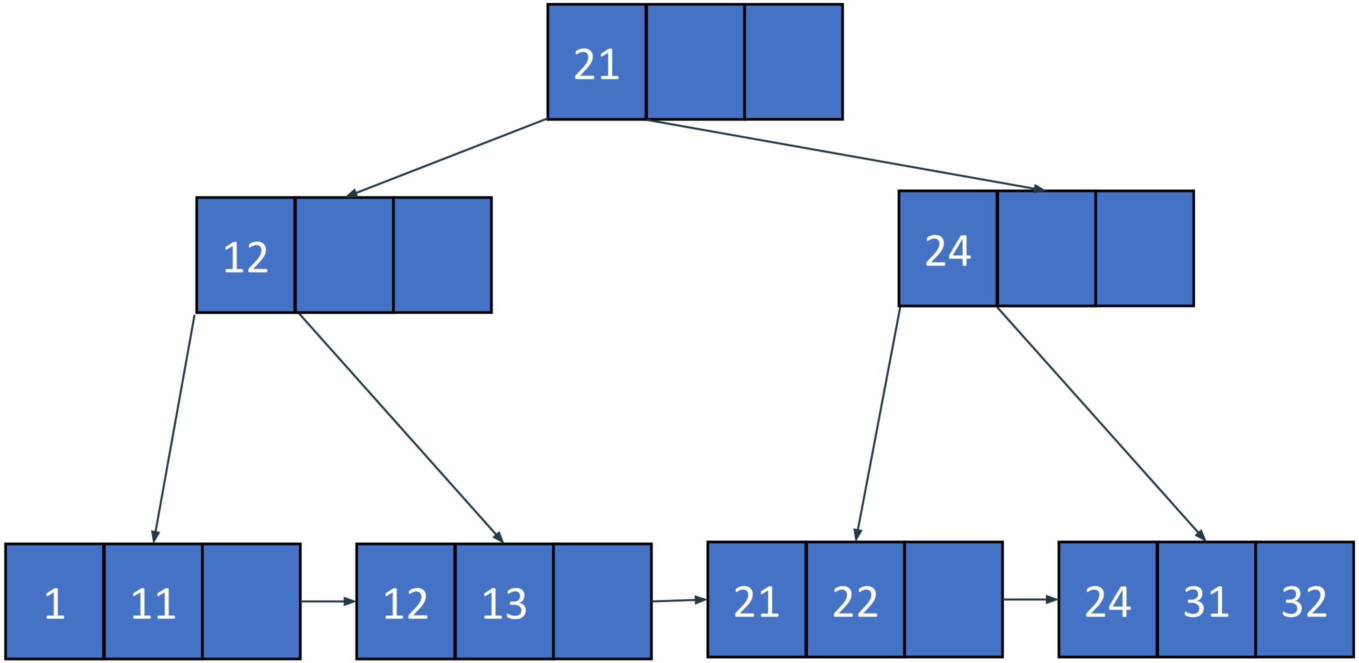
   ii. Which key will first make the table double the size? ⇒ **10**

a. Insert 24 to the following B+tree

b. Delete 23 from the B+tree obtained in part a.



a. Insert 24 to the following B+tree
b. Delete 23 from the B+tree obtained in part a.



- Consider the relations **R(A, B)**, **S(C, D)** and **T(D, B, E)** with keys underlined, i.e., attribute A is a key for R, C is a key for S, and D is a key for T.
- **Tuples from S and T take up 50 bytes**, while **tuples from R comprise 40 bytes**. Values for attribute **S.D take up 10 bytes**.
- The **page size is set to 4000 bytes** and there are **8 pages in the buffer.**
- The statistics show that **S contains 36000 tuples**; that **T contains 10000 tuples**; that **R contains 12000 tuples**.
- Furthermore, values in **S.D are uniformly distributed in the range [0,100],** values in T**.E are uniformly distributed in the range [10000 , 30000]**, and **T has 5000 distinct values for attribute B.** In addition, the following **histogram information is available for R.B**:

| range | number of tuples |
|---|---|
| [0,19] | 5000 |
| [20,39] | 200 |
| [40,59] | 800 |
| [60,79] | 2000 |
| [80,100] | 4000 |

- Moreover, relation S has a clustered B+-tree index on attribute D; relation T has unclustered hash indexes on attributes B and E (separately).
- Assume that we only consider **left-deep plans**, and two join algorithms (**block nested-loops**, and **sort-merge join**) only. Use **Selinger Optimizer (dynamic programing algorithm)** to find the optimal cost-based query-plan to the following SQL query (i.e. find the best query tree and the algorithm associated with each operator in the tree).

- **SELECT R.A, S.C, T.D FROM R,S, T**
  **WHERE R.B==30 AND S.D <=60 AND T.E != 10000 AND R.B == T.B AND S.D == T.D**

$$\pi_{R.A,S.C,T.D}\left((\sigma_{R.B=30}(R)) \bowtie_{R.B==T.B} (\sigma_{T.E\,!=10000}(T)) \bowtie_{S.D==T.D} (\sigma_{S.D\leq 60}(S))\right)$$

**The best way to select one relation**

$\pi_{R.A,S.C,T.D}\ ((\sigma_{R.B=30}(R)) \bowtie_{R.B==T.B} (\sigma_{T.E\ !=10000}(T)) \bowtie_{T.D==S.D} (\sigma_{S.D\leq 60}(S)))$

- $\sigma_{R.B=30}(R)$
  - Full table scan to search the values (no index nor unique values)
    = (number of tuples) * (record size) /(page size)
    = 12000 * 40 / 4000 = 120 IOs
  - 200 tuples [20,39] **(from givens)**
  - Cardinality = 1/20 * 200 = 10 tuples = 400 bytes = 1 block
- $\sigma_{T.E\ !=10000}(T)$
  - Hash indexes are not useful for inequality predicates.
  - Full table scan = (number of tuples) * (record size) / (page size)
    = 10000 * 50 / 4000 = 125 IOs
  - T.E are uniformly distributed in the range [10000 , 30000] **(from givens)**
  - Cardinality = 20000/20001 * 10000 = 10000 tuples = 500000 bytes = 125 blocks
- $\sigma_{S.D\leq 60}(S)$
  - #IOs = index access (using the B+tree index) + fetch blocks
  - S.D are uniformly distributed in the range [0,100] **(from givens)**
  - Cardinality = 61/101 * 36000 = 21743 tuple = 1087150 bytes = 272 blocks
  - Index record ⇒ D key (10 bytes) + pointer (assume 10 bytes) ⇒ 20 bytes
    ⇒ 4000 / 20 = 200 index entry / page
  - The relation can fit into 36000*50/4000 = 450 block
  - First level = ceil(450 / 200) = 3 pages
  - Second level have only 3 pointers to the first level ⇒ 2 level index is needed
  - #IOs = 2 + 272 = 274 IOs

$\pi_{R.A,S.C,T.D}$ $((\sigma_{R.B=30}(R)) \bowtie_{R.B==T.B} (\sigma_{T.E\ !=10000}(T)) \bowtie_{T.D==S.D} (\sigma_{S.D\leq 60}(S)))$

**The best way to select two relations**

- $(\sigma_{R.B=30}(R)) \bowtie_{R.B==T.B} (\sigma_{T.E\ !=10000}(T))$
  - Nested loop join
    - Cost = M + ceil( M / (B-2) ) * N ⇒ (M is the smaller relation)
      = 1+ ceil(1/6) * 125 = 126 IOs
  - Sort merge join
    - Cost = 2M(1+ ceil($\log_{B-1}$(ceil(M/B))) + 2N(1+ ceil($\log_{B-1}$(ceil(N/B))) + M + N (higher cost)
  - Total cost = current cost + sub-problems cost = 126 + 120 + 125 = 371 IOs
  - Cardinality = 10 * 10000 / max(1,5000(number of distinct values in T.B) ) = 20 tuples = 1 block

- $(\sigma_{T.E\ !=10000}(T)) \bowtie_{T.D==S.D} (\sigma_{S.D\leq 60}(S))$
  - Nested loop join
    - Cost = M + ceil( M / (B-2) ) * N ⇒ (M is the smaller relation)
      = 125 + ceil((125)/(8-2))*272 = 5837 IOs
  - Sort merge join
    - Note that the relation S is sorted on the attribute D ⇒ Ignore sorting S
    - Cost = 2M(1+ ceil($\log_{B-1}$(ceil(M/B))) + ~~2N(1+ ceil($\log_{B-1}$(ceil(N/B))) +~~ M + N
      = 2 *125 * 3 + ~~2 * 272 * 3 +~~ 125 + 272 = 1147 IOs
  - Total cost = current cost + sub-problems cost = 1147 + 274 + 125 = 1546 IOs
  - Cardinality = 21743 * 10000 / max(61,10000 ) = 21743 tuples
    ⇒ 21743 * 100 / 4000 = 544 blocks

$\pi_{R.A,S.C,T.D}$ $((\sigma_{R.B=30}(R)) \bowtie_{R.B==T.B} (\sigma_{T.E\ !=10000}(T)) \bowtie_{T.D==S.D} (\sigma_{S.D\leq 60}(S)))$

**The best way to select three relations**

**Continue by yourself**

- $((\sigma_{R.B=30}(R)) \bowtie_{R.B==T.B} (\sigma_{T.E\ !=10000}(T))) \bowtie_{T.D==S.D} (\sigma_{S.D\leq 60}(S)$
  - Nested loop join
    - Cost = M + ceil( M / (B-2) ) * N ⇒ (M is the smaller relation)
      =
  - Sort merge join
    - Cost = $2M(1+ ceil(log_{B-1}(ceil(M/B))) + 2N(1+ ceil(log_{B-1}(ceil(N/B))) + M + N$
      =
  - Total cost = current cost + sub-problems cost =

- $((\sigma_{T.E\ !=10000}(T)) \bowtie_{T.D==S.D} (\sigma_{S.D\leq 60}(S))) \bowtie_{R.B==T.B} (\sigma_{R.B=30}(R)$
  - Nested loop join
    - Cost = M + ceil( M / (B-2) ) * N ⇒ (M is the smaller relation)
      =
  - Sort merge join
    - Cost = $2M(1+ ceil(log_{B-1}(ceil(M/B))) + 2N(1+ ceil(log_{B-1}(ceil(N/B))) + M + N$
      =
  - Total cost = current cost + sub-problems cost =