Sheet ⑥

Name: Asmaa Gamal Abdel-Halem Mabrouk Nagy

Course: Database « Audience Course »

Department: Communications & Electronics

DBMS

Query Processing & External Sorting

1 (a) 1- Conjunctive selection: Condition AND Condition

ex

```
SELECT * FROM EMPLOYEE
WHERE Department = 'IT' AND Salary > 5000;
```

2- Disjunctive Selection: OR

ex

```
SELECT * FROM PRODUCT
WHERE Category = 'Electronics' OR Price < 1000;
```

(b) Multiple options for Execution:

* index scan
* table "

] or → both

∴ while < Conjuctive / disjunctive > Selection queries express logical conditions to filter data, the actual execution can vary based on the data engine's optimization, indices, & information.

[2] Alternative ways

① * "GROUP BY" : this query groups the data by the specified attribute and provides the count of accurrences for each unique value.

② * Aggregate fun:

ex: MIN( ), MAX( ), AVG( )

to get info about each unique value.

③ * window fun: ex: ROW_NUMBER( )

to assign a unique num to each row within a Partition, then, you can filter based on this num

④ * JOIN.

[3] Implementation steps

① scan the Data
② Grouping (if required)
③ initialization of aggregate Variables
④ iterating " "
⑤ update the "
⑥ finalizing & outputting Results.
⑦ optimization using
- index usage
- caching
- Parallel Processing
- Query optimization

4] * outer Join implementation :

1- Nasted Loop Join.
2. Hash Join.
3. Merge Join.
4. left & right outer Join.

---

* Non-Equi Join :

1. theta Join
2. Nested Loop Join with Non-equi Condition.
3. Hash-Join] With → " " "
4. Merge-Join]
5. Fitering Post Join

5] * iterator concept: is a Programming Pattern to traVerse a Container without exposing the underlying details of its implementation

* iterator Methods

'next()' → or their equivalent fun. ← 'hasNext()'

6] * Sparse index "non-dense" Can be used to implement aggregate Operator, but there are Some issues: ① inComplete Coverage
② Multiple index Look ups I/O ↑
③ optimization Challenges.

[7] Steps of: LEFT OUTER JOIN using Sort-Merge Join algorithm

1- Sort the left & right table, based on 'Join-k'
2- initialize Pointers 'left Pointer = 0', 'right-Pointer = 0'
3- Perform merge
4- Process Remaining Rows in the Left table

[8] Using External Sorting

a 10,000 Page, ③ buffers:

(i) $\#\text{runs}_{1^{st}\text{ Pass}} = \left\lceil \frac{\text{tot Pages}}{\text{buffer Pages}} \right\rceil = \left\lceil \frac{10,000}{3} \right\rceil = 3,333$ runs

(ii) $\#\text{ Passes} = \log_{B-1}(R) = \log_{2}(3333) \simeq 11$ Passes

(B-1: buffer; R: Runs)

(iii) tot I/O Cost = $\#\text{Pages}_{tot} * \#\text{ Passes} = 10,000 * 11 = 110\,000$ I/O

(iv) $\#$ Buffer Pages in ② Passes $= \#\text{buffer}_{@\text{each Pass}} * 2 = 6$ buffer Pages

b 20,000 Page & ⑤ buffers:

$\#\text{runs} = \frac{20,000}{5} = 4,000$ runs.

$\#\text{ Passes} = \log_{4} 4000 \simeq 6$ Passes

$\#\text{ I/O} = 20,000 * 6 = 120\,000$ I/O

$\#$ Buffers in ② Passes $= 5 * 2 = 10$ buffers.

Continue of Q(8):

(c) 2 Million Pages & 17 buffers :

$$\#\text{ runs} = \frac{2\,000\,000}{17} \simeq 117647 \text{ runs}$$

$$\#\text{ Passes} = \log_{16} 117647 = 4 \text{ Passes}$$

$$\#\text{ I/O} = 2^{M} * 4 = 8{,}000{,}000 \text{ I/O}$$

$$\#\text{ Buffers in (2) Passes} = 17 * 2 = 34 \text{ buffer}$$