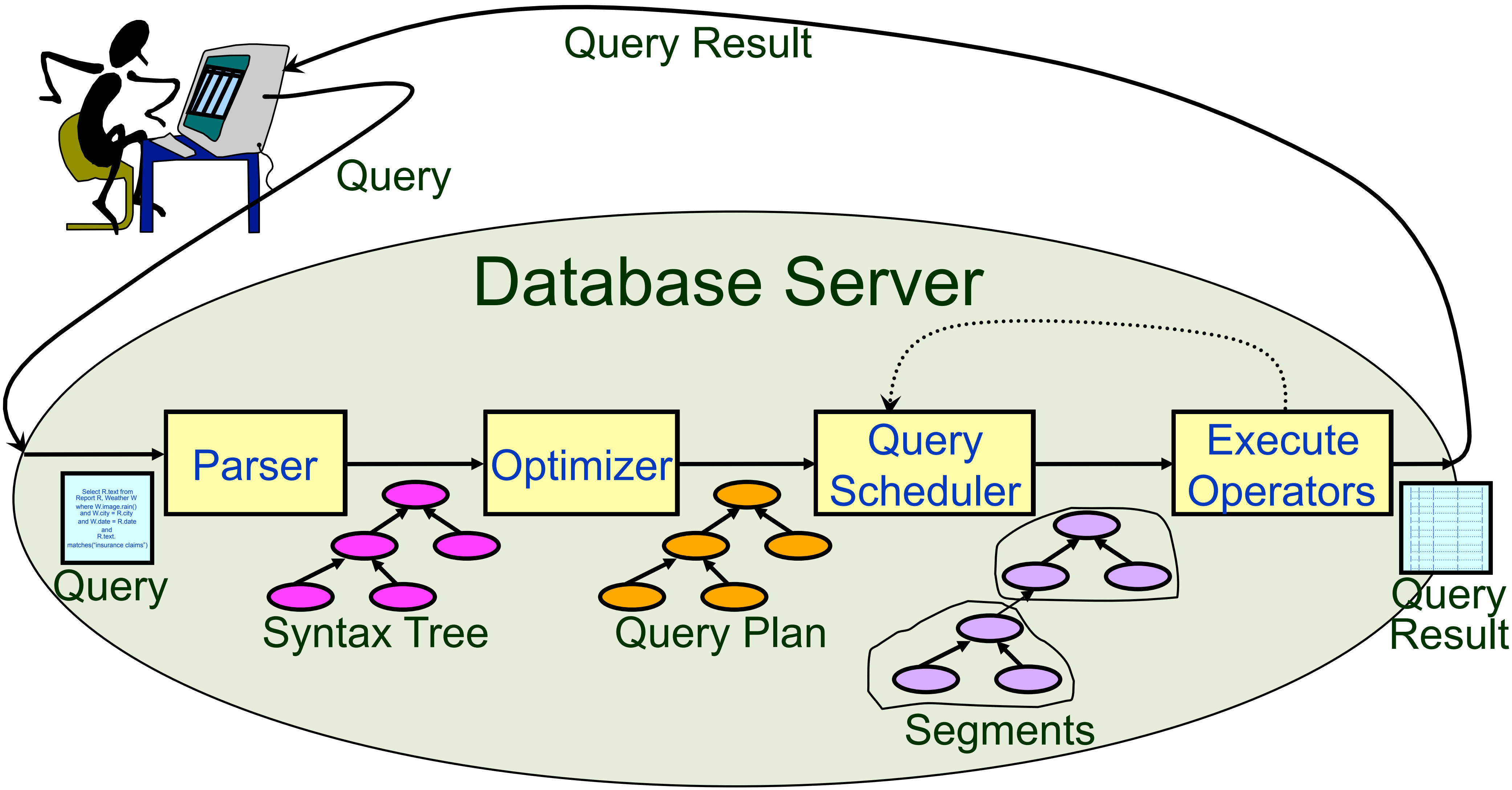
Fall 2013

QUERY PROCESSING

[LOOSELY BASED ON CH 12.1-12.3 AND 14 IN THE COW BOOK]

Life Cycle of a Query



Problem Statement

```
CREATE TABLE User (
   uid          INTEGER,       -- unique id or the user
   login        VARCHAR(20)    -- unique login name
   lname        VARCHAR(80),   -- lastname
   fname        VARCHAR(80),   -- firstname
   dob          DATE,          -- date of birth
   PRIMARY KEY (uid),          -- primary key for the table
   UNIQUE (login)              -- twid is also unique
);


CREATE TABLE Messages (
   uniqueMsgID INTEGER,        -- unique message id
   tstamp      TIMESTAMP,      -- when was the message posted
   uid         INTEGER,        -- unique id of the user
   msg         VARCHAR (140),  -- the actual message
   zip         INTEGER,        -- zipcode when the message was posted
   reposted    BOOLEAN         -- is this a reposted message?
   PRIMARY KEY (uniqueMsgID),  -- primary key
   FOREIGN KEY (uid) REFERENCES USER – Foreign key to the User table
   );
```

Problem Statement

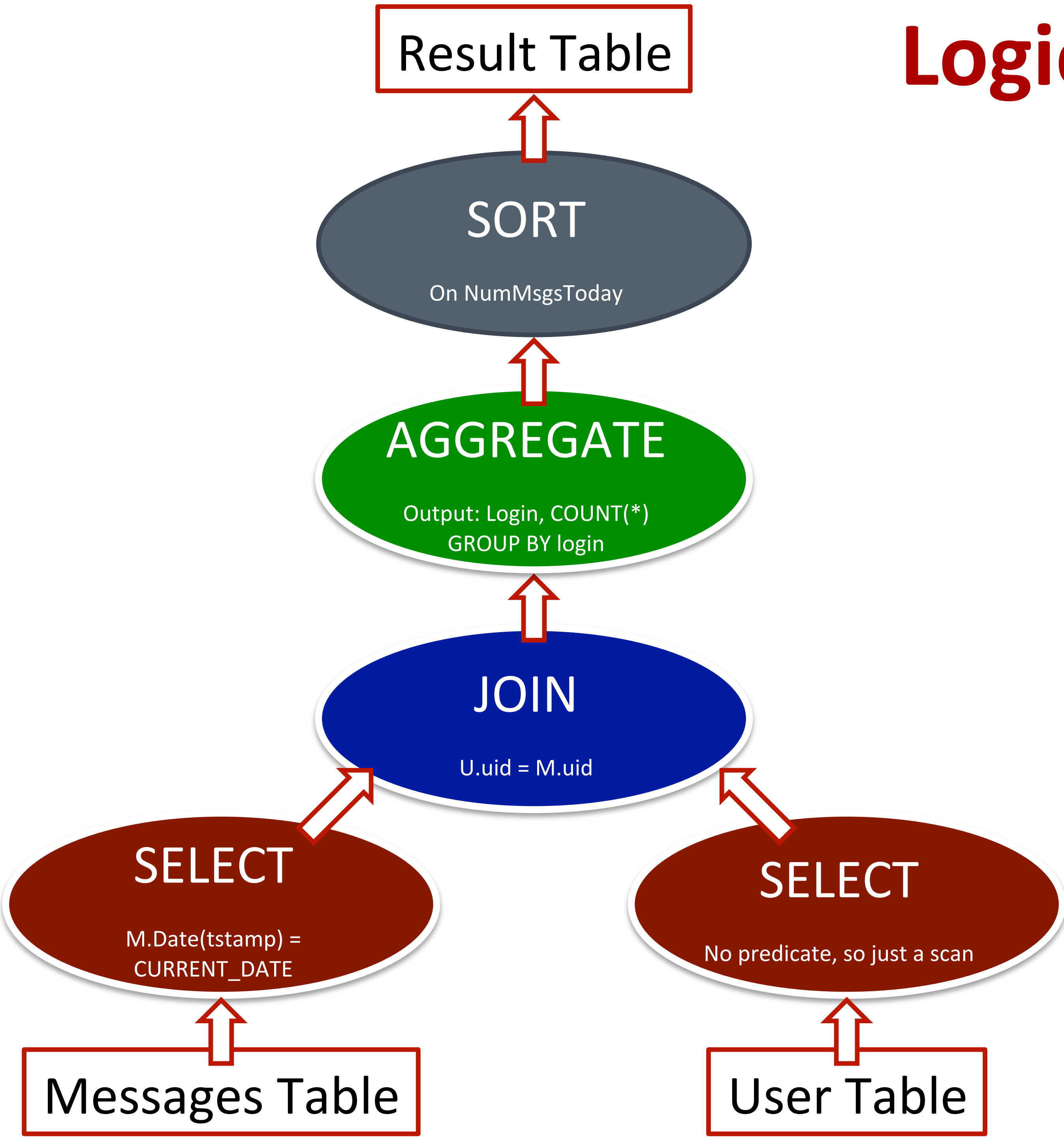
- Run the following query:

```
SELECT U.login AS login, COUNT(*) AS NumMsgsToday
FROM   User U, Messages M
WHERE  U.uid = M.uid
  AND  M.Date(tstamp) = CURRENT_DATE  -- select msgs posted today
GROUP BY U.login                      -- group by login
ORDER BY NumMsgsToday DESC            -- order by descending msg count
```

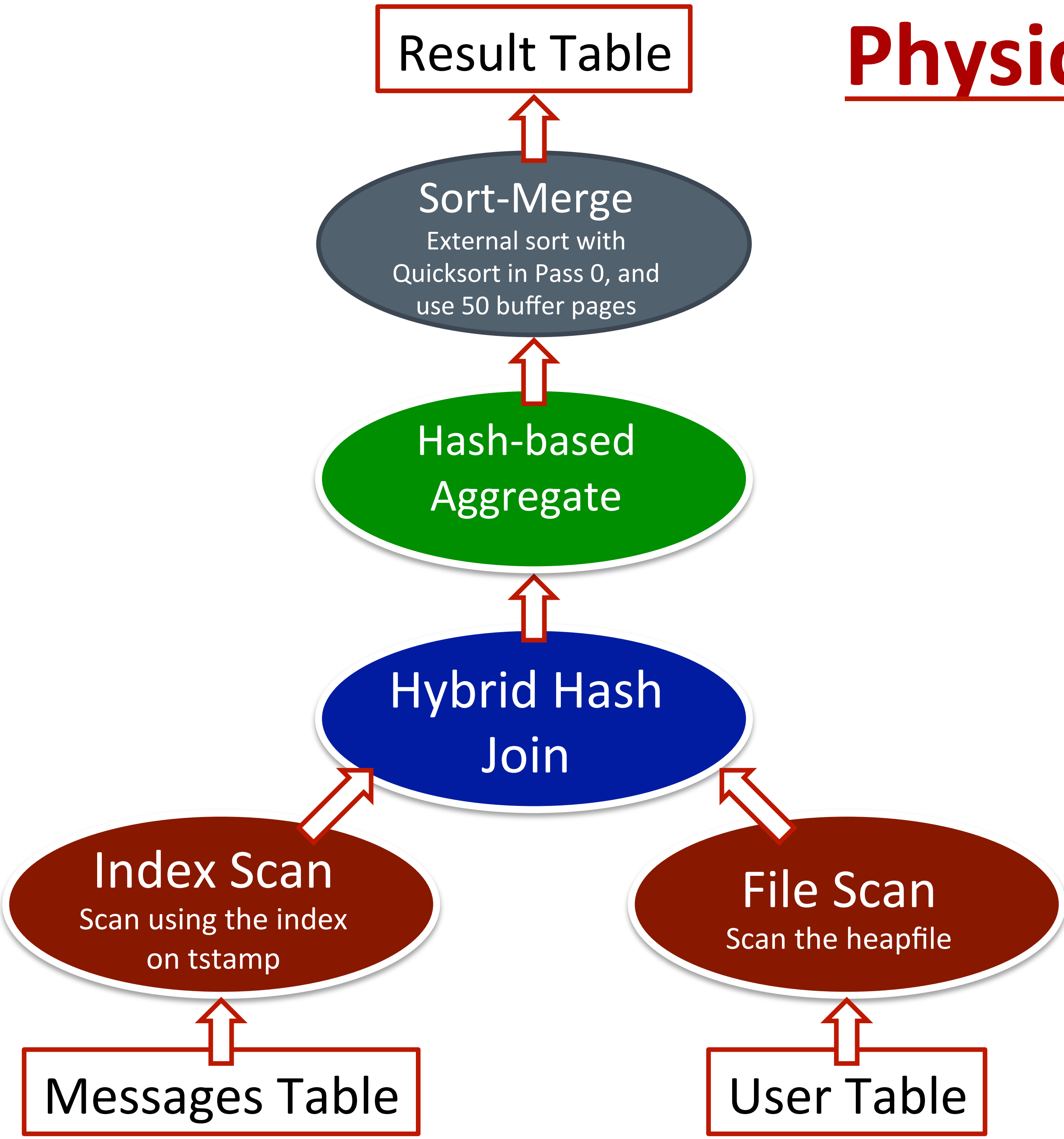
Sample output table

login	NumMsgsToday
angelak	211
jackdr	101
petescafe	10
…	…

Logical Query Plan



Physical Query Plan



Select Operation

- Algorithms: File Scan or Index Scan
- **File Scan:** Disk I/O cost:
- **Index Scan:** (on some predicate). Disk I/O cost:
  - Hash: O( ) can only use with equality predicates
  - B+-tree: O( ) + X
    - X = number of selected tuples/number of tuples per page
    - X = 1 per selected tuple with an unclustered index. To reduce the value of X, we could sort the rids and then fetch the tuples.
  - Bitmap Index:

When to use a B+tree index

- Consider
  - A relation with 1M tuples
  - 100 tuples on a page
  - 500 (key, rid) pairs on a page

data pages
= 1M/100 = 10K pages
leaf idx pgs
= 1M / (500 * 0.67)
~ 3K pages

	1% Selection	10% Selection
Clustered	30 + 100	300 + 1000
Non-Clustered	30 + 10,000	300 + 100,000
NC + Sort Rids	30 + (~ 10,000)	300 + (~ 10,000)

⇨ Choice of Index access plan, consider:
1. Index Selectivity **2. Clustering**
⇨ Similar consideration for hash based indices

When can we use an index

- Notion of “index matches a predicate”
- Basically mean when can an index be used to evaluate predicates in the query

General Selection Conditions

- Index on (R.a, R.b)
  - Hash or tree-based
- Predicate:
  - R.a > 10
  - R.b < 30
  - R.a = 10 and R.b = 30
  - R.a = 10 or R.b = 30
- Predicate: (p1 and p2) or p3
- Convert to Conjunctive Normal Form(CNF)
  (p1 or p3) and (p2 or p3)
- An index ***matches*** a predicate
  - Index can be used to evaluate the predicate

Index Matching

- B+-tree index on <a, b, c>
  - a=5 and b= 3?
  - a > 5 and b < 3
  - b=3
  - a=7 and b=5 and c=4 and **d>4**
  - a=7 and c=5

- •
- •
- •
- • **(primary conjunct)**
- •

Hash Idx

- •
- •
- •
- •
- •

- Index matches (part of) a predicate
  1. Conjunction of terms involving only attributes (no disjuctions)
  2. Hash: only equality operation, predicate has all index attributes.
  3. Tree: Attributes are a prefix of the search key, any ops.

Index Matching

- A predicate could match more than 1 index
- Hash index on <a, b> and B+tree index on <a, c>
  - a=7 and b=5 and c=4   Which index?

  - Option1: Use either (or a file scan!)
    - Check selectivity of the primary conjuct
  - Option2: Use both! Algorithm: Intersect rid sets.
    - Sort rids, retrieve rids in both sets.
    - Side-effect: tuples retrieved in the order on disk!

Selection

- Hash index on <a> and Hash index on <b>
  - a=7 **or** b>5 Which index?
  - Neither! File scan required for b>5
- Hash index on <a> and B+-tree on <b>
  - a=7 **or** b>5 Which index?
  - Option 1: Neither
  - Option 2: Use both! Fetch rids and union
    - Look at selectivities closely. Optimizer!
- Hash index on <a> and B+-tree on <b>
  - (a=7 **or** c>5) and b > 5 Which index?
  - Could use B+-tree (check selectivity)

Projection Algorithm

- Used to project the selected attributes.

Simple case: Example SELECT R.a, R.d.

  – Algorithm: for each tuple, only output R.a, R.d

Harder case: DISTINCT clause

- Example: SELECT DISTINCT R.a, R.d
  – Remove attributes <u>and</u> eliminate duplicates
- Algorithms
  – Sorting: Sort on <u>all</u> the projection attributes
    - Pass 0: eliminate unwanted fields. Tuples in the sorted-runs may be smaller
    - Eliminate duplicates in the merge pass & in-memory sort
  – Hashing: Two phases
    - Partitioning
    - Duplicate elimination

Hashing

Can h1 = h2?

What if the hash table for a partition overflows, i.e. can't fit in memory?

$R' = \pi_P(R)$

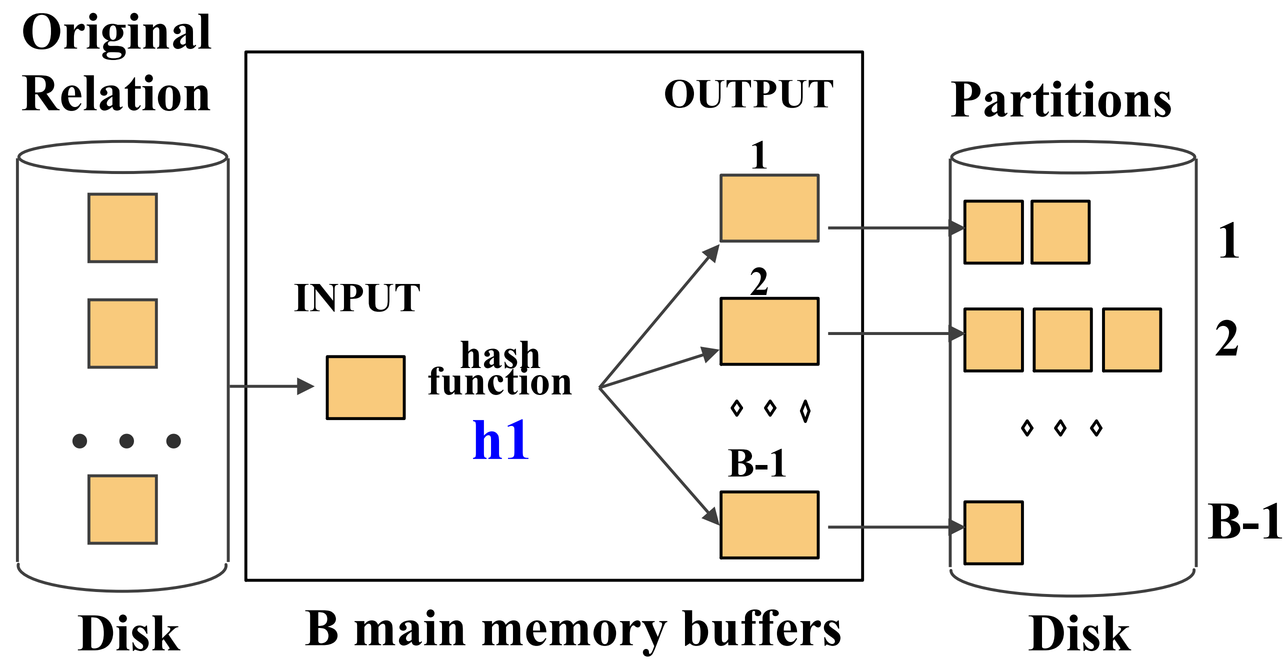
No overflows if

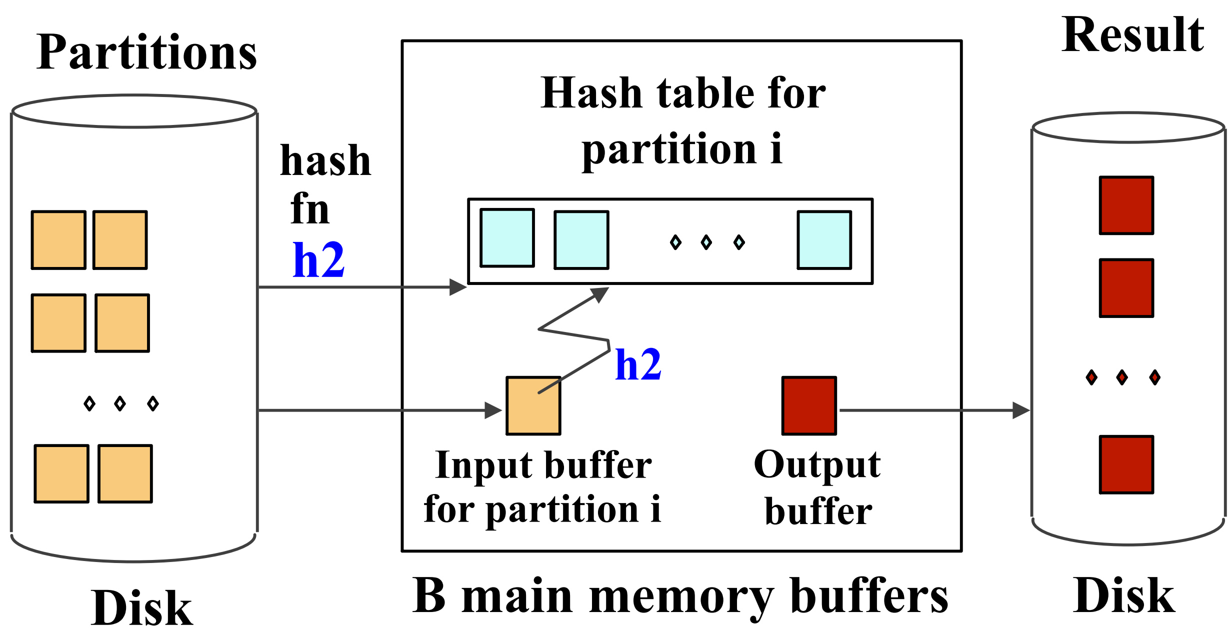
$|R'| < B^2/F$

F = fudge factor (to account for the hash table)

Part. Sz, $P = |R'|/B-1$

Hash Tab Sz $= F*P < B$





Projection …

- Sort-based approach
  - better handling of skew
  - result is sorted
  - I/O costs are comparable if $B^2 > |R'|$
- Index-only scan
  - Projection attributes subset of index attributes
  - Apply projection techniques to data entries (much smaller!)
- If an ordered (i.e., tree) index contains all projection attributes as *prefix* of search key:
  1. Retrieve index data entries in order
  2. Discard unwanted fields
  3. Compare adjacent entries to eliminate duplicates (if required)