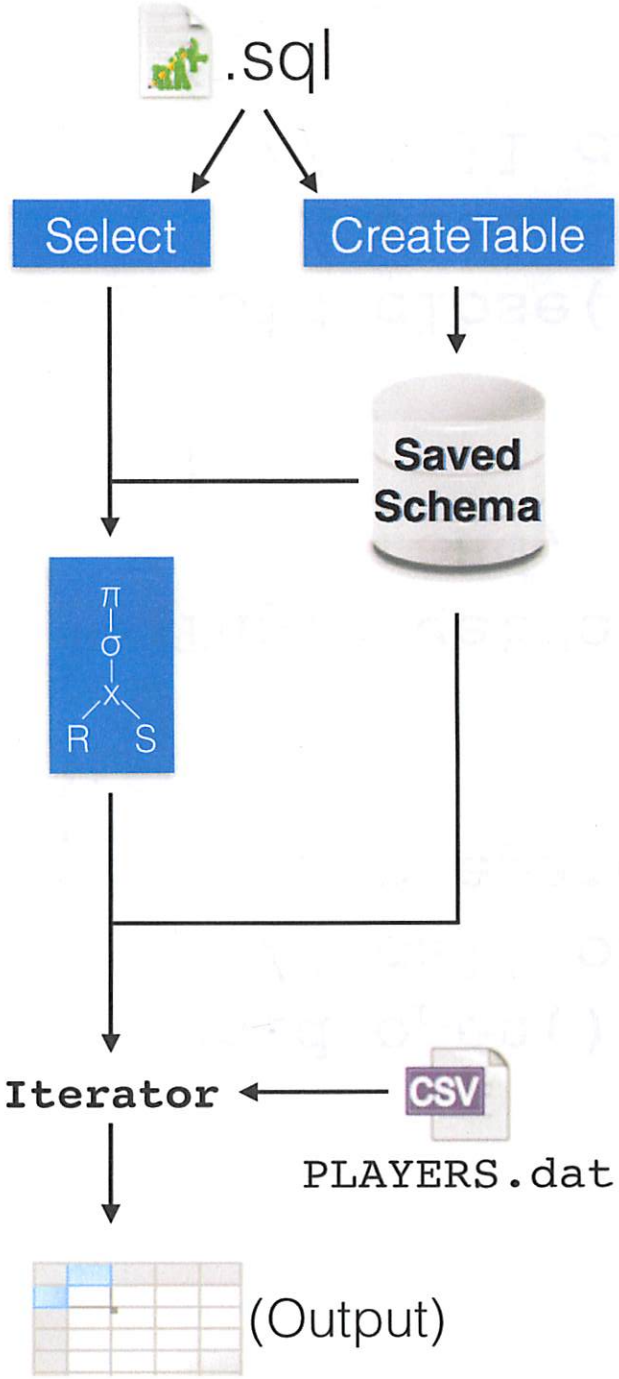
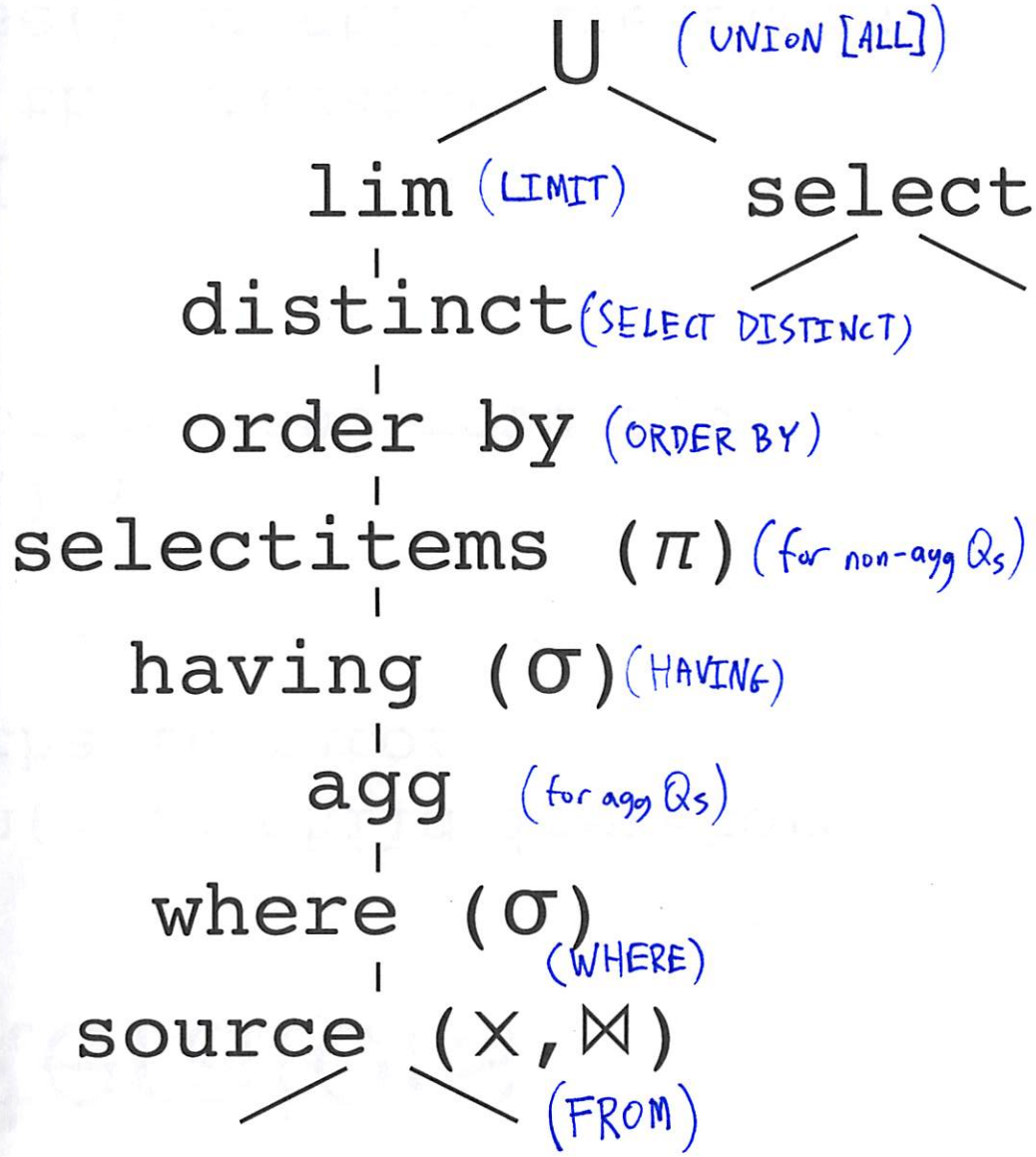
Exam Review

SQL → RA → Iter


.sql
Select
CreateTable
Saved Schema
π
σ
X
R
S
Iterator
CSV
PLAYERS.dat
(Output)


U
(UNION [ALL])
lim (LIMIT)
select
distinct (SELECT DISTINCT)
order by (ORDER BY)
selectitems (π) (for non-agg Qs)
having (σ) (HAVING)
agg (for agg Qs)
where (σ)
(WHERE)
source (X, ⋈)
(FROM)

Iterators

```
void open() {
  // call open() on child iterators
  // prepare the iterator
}

Tuple getNext() {
  // read, process, and return a tuple
}

void close() {
  // clean-up the iterator
  // call close() on child iterators
}
```

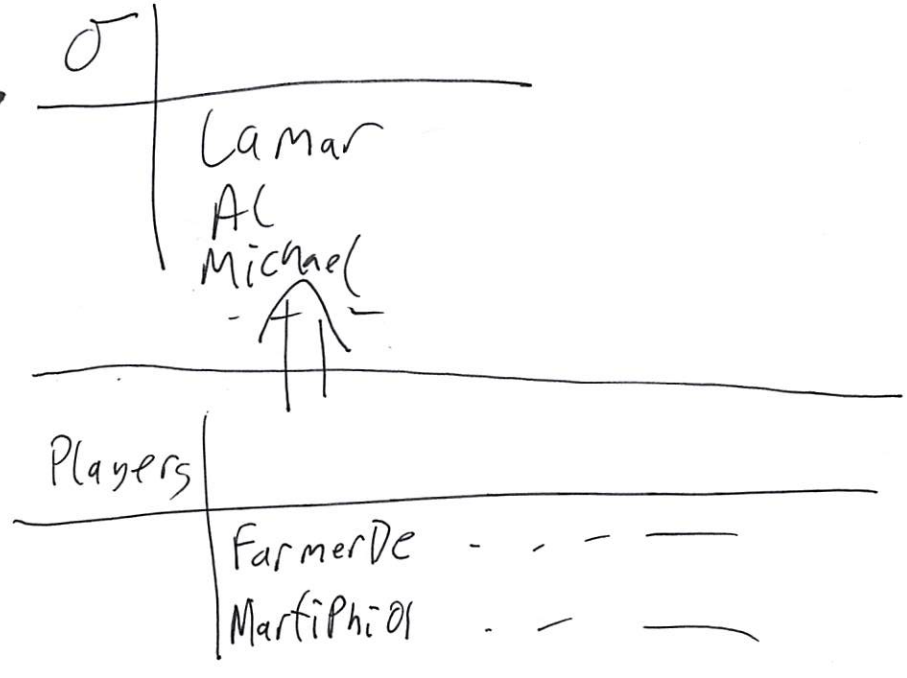
```
CREATE TABLE PLAYERS(
    ID string, FIRSTNAME string, LASTNAME string,
    FIRSTSEASON int, LASTSEASON int, WEIGHT int,
    BIRTHDATE date
);

INSERT INTO
  PLAYERS(ID, FIRSTNAME, LASTNAME, FIRSTSEASON, LASTSEASON,
                                                  WEIGHT, BIRTHDATE)
VALUES
  ('FarmeDe01', 'Desmon',  'Farmer',  2006, 2008, 220, 1981-10-07)
  ('MARTIPH01', 'Phil',    'Martin',  1954, 1954, 190, 1928-04-02)
  ('GREENLA01', 'Lamar',   'Green',   1969, 1974, 210, 1947-03-22)
  ('BIANCAL01', 'Al',      'Bianchi', 1956, 1965, 185, 1932-03-26)
  ('JORDAMI01', 'Michael', 'Jordan',  1984, 2002, 195, 1963-02-17)
  ('ROBERAN01', 'Anthony', 'Roberts', 1977, 1983, 185, 1955-04-15)
  ('SMITHTO01', 'Tony',    'Smith',   1990, 2000, 185, 1968-06-14);

-----------------------------------------------------------

SELECT FIRSTNAME, LASTNAME, FIRSTSEASON, LASTSEASON
FROM PLAYERS
WHERE LASTSEASON-FIRSTSEASON>5;

-----------------------------------------------------------

SELECT P1_FIRSTNAME, P2_FIRSTNAME
FROM (
  SELECT P1.FIRSTNAME AS P1_FIRSTNAME, P2.FIRSTNAME AS P2_FIRSTNAME
         P1.FIRSTSEASON AS P1_FIRSTSEASON, P1.LASTSEASON AS
P1_LASTSEASON,
         P2.FIRSTSEASON AS P2_FIRSTSEASON, P2.LASTSEASON AS
P2_LASTSEASON
    FROM PLAYERS P1, PLAYERS P2
  WHERE P1.ID<>P2.ID
  ) SUB_Q
WHERE P1_FIRSTSEASON<P2_FIRSTSEASON
  AND P1_LASTSEASON>P2_LASTSEASON;

-----------------------------------------------------------

SELECT EXPERIENCE, COUNT(*)
FROM (
  SELECT ID, FIRSTNAME, LASTNAME,
         (LASTSEASON-FIRSTSEASON) AS EXPERIENCE
  FROM PLAYERS
)
GROUP BY EXPERIENCE;
```

```
SELECT FIRSTNAME, LASTNAME,
       FIRSTSEASON, LASTSEASON
FROM PLAYERS
WHERE LASTSEASON-FIRSTSEASON>5;
```

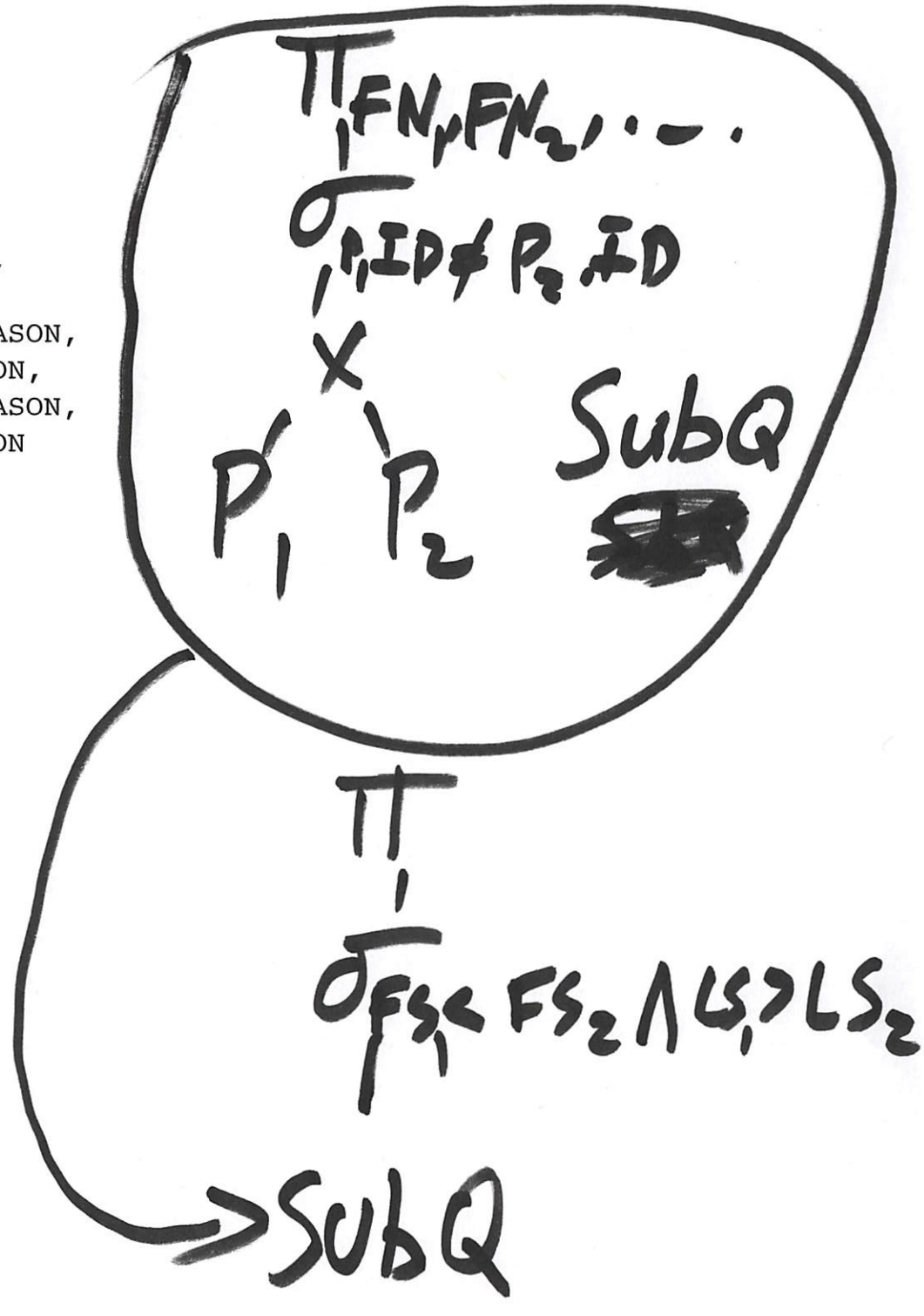
$\pi_{FN, LN, FS, LS}$

$\sigma_{LS-FS \geq 5}$

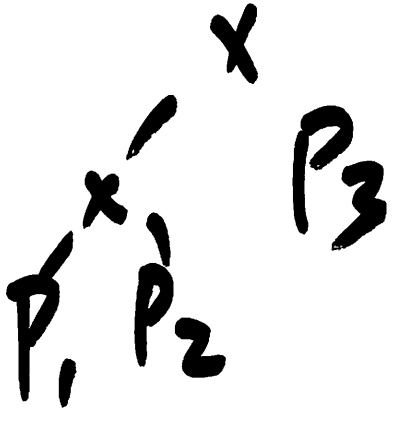
Players



```sql
SELECT P1_FIRSTNAME, P2_FIRSTNAME
FROM (
  SELECT P1.FIRSTNAME AS P1_FIRSTNAME,
         P2.FIRSTNAME AS P2_FIRSTNAME,
         P1.FIRSTSEASON AS P1_FIRSTSEASON,
         P1.LASTSEASON AS P1_LASTSEASON,
         P2.FIRSTSEASON AS P2_FIRSTSEASON,
         P2.LASTSEASON AS P2_LASTSEASON
    FROM PLAYERS P1, PLAYERS P2
  WHERE P1.ID<>P2.ID
  ) SUB_Q
WHERE P1_FIRSTSEASON<P2_FIRSTSEASON
  AND P1_LASTSEASON>P2_LASTSEASON;
```

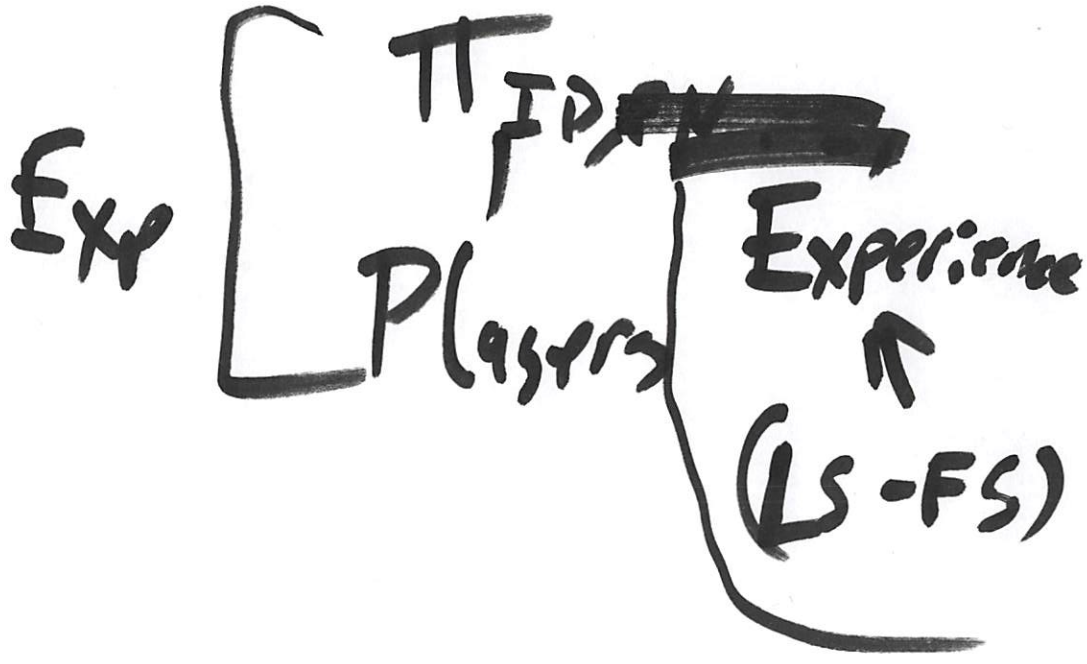


$(P_1 \times P_2) \times P_3$

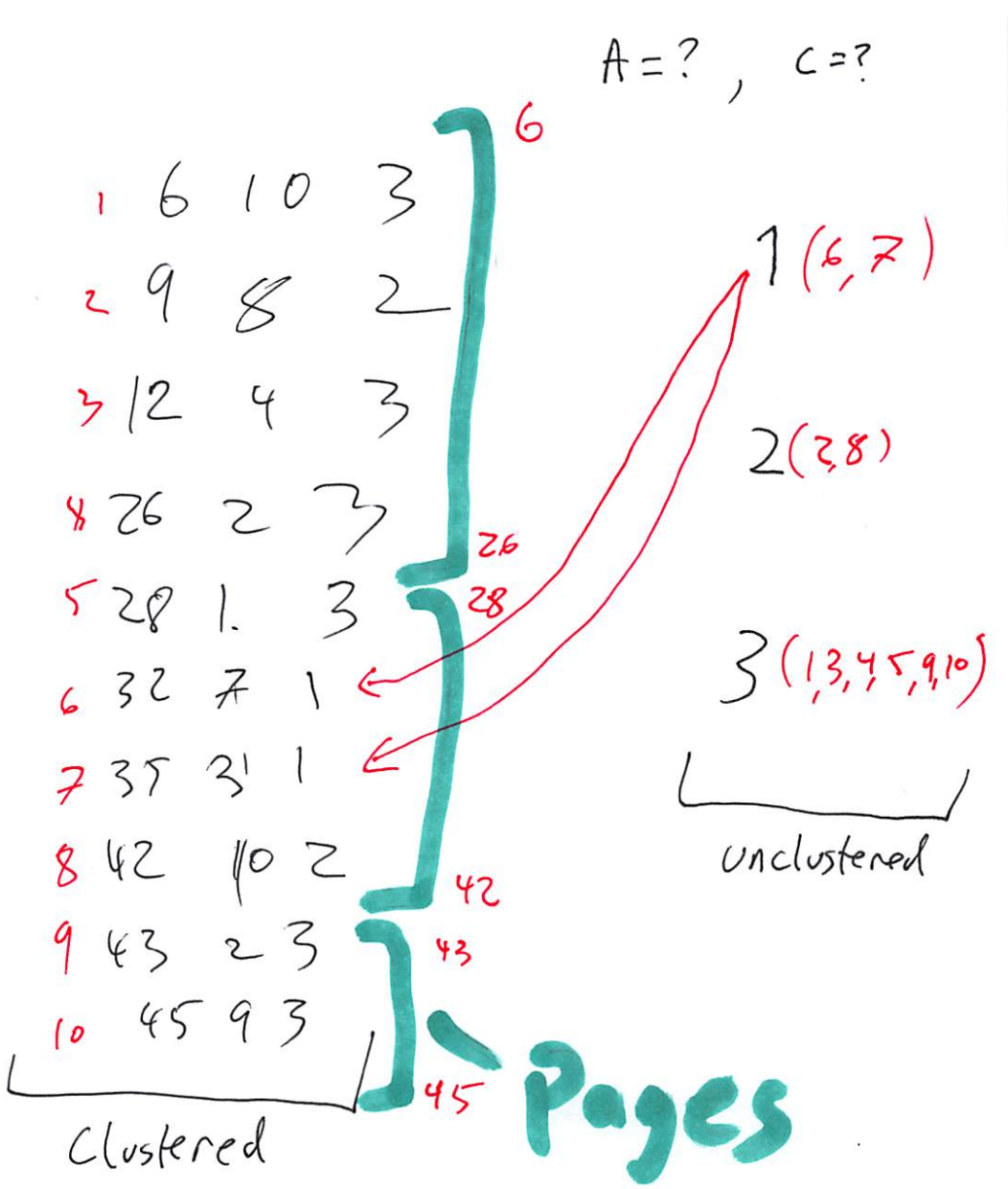


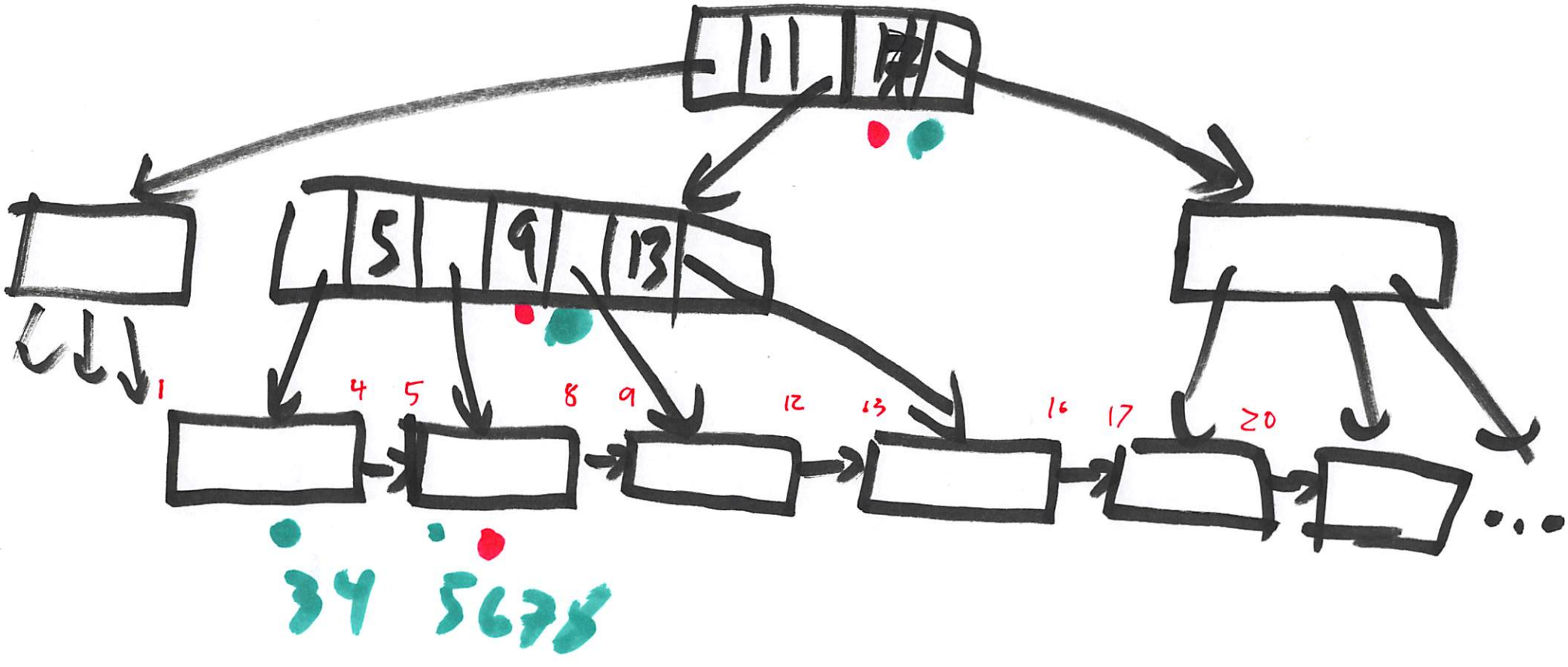
Aggregate

```
SELECT EXPERIENCE, COUNT(*)
FROM (
  SELECT ID, FIRSTNAME, LASTNAME,
         (LASTSEASON-FIRSTSEASON)
             AS EXPERIENCE
  FROM PLAYERS
) Exp
GROUP BY EXPERIENCE;
```



R	A	B	C
1	6	10	3
7	35	3	1
4	43	2	3
3	12	4	3
8	42	10	2
5	28	1	3
6	32	7	1
2	9	8	2
4	26	2	3
10	45	9	3





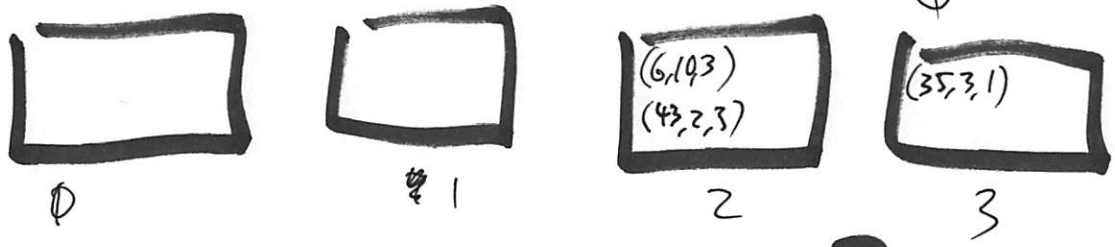
3 4 5 6 7 8

- Read 7
- Scan 3 - 8

Find 7

$h(7) = 3$

organize By B



$$h(k) = k \% B$$

B
↑
4

IO Cost

	HeapFile	Sorted File	BtTree	Hash
Lookup A=?	N	$Log_2 N$	$Log_k N$	1
Scan A<40 ∧ A>3	N		$Log_k N$ + # Pages	N
Insert				
Delete				

R	A	B
page1	1	2
	3	4
	5	6
page2	7	8
	9	10
	11	12

S	B	C
page1	12	8
	12	9
	2	2
	2	7
page2	4	4
	6	2
	8	6

$R \bowtie_B S$

NLJ

$\forall R \rightarrow \forall S$

<1,2> → <12,8> 2≠12 → ∅
→ <12,9> 2≠12 → ∅
→ <2,2> 2=2 → <1,2,2,2>
→ <2,7> 2=2 → <1,2,2,7>
→ <4,4> 2≠4 → ∅
⋮

<3,4> → <12,8> 4≠12 → ∅
→ <12,9> 4≠12 → ∅
⋮

<5,6> → ⋮

GHJ

Hash S, Stream R

S	B	<B, C>
	12	→ <12,8> <12,9>
	2	→ <2,7> <2,2>
	4	→ <4,4>
	6	→ <6,2>
	8	→ <8,6>

1,2 → {<2,7>,<2,2>}
⋮

BNLJ

[1,2 / 3,4 / 5,6] → [12,8 / 12,9 / 2,2 / 2,7]

→ <1,2> → <12,8> 2≠12 → ∅
→ <12,9> 2≠12 → ∅
→ <2,2> 2=2 → <1,2,2,2>
→ <2,7> 2=2 → <1,2,2,7>

→ <3,4> → <12,8> 3≠12 → ∅
→ <12,9> 3≠12 → ∅
→ <2,2> 3≠2 → ∅
→ <2,7> 3≠2 → ∅

→ <5,6> → <12,8> 6≠12 → ∅
→ <12,9> 6≠12 → ∅
→ <2,2> 6≠2 → ∅
→ <2,7> 6≠2 → ∅

→ [4 4 / 6 2 / 8 6] → 1,2 → 4,4 → 2≠4 → ∅
→ 6,2 → 2≠6 → ∅
→ 8,6 → 2≠8 → ∅

⋮

$R \bowtie_B S$

SMJ

$sort_B(R)$	A	B
	1	2
	3	4
	5	6
	7	8
	9	10
	11	12

$sort_B(S)$	B	C
	2	2
	2	7
	4	4
	6	2
	8	6
	12	8
	12	9

2 $[1,2] \times \begin{bmatrix}2,2\\2,7\end{bmatrix}$ → $\langle 1,2,2,2\rangle$ $\langle 1,2,2,7\rangle$

4 $[3,4] \times [4,4]$ → $\langle 3,4,4,4\rangle$

6 $[5,6] \times [6,2]$ → $\langle 5,6,6,2\rangle$

8 $[7,8] \times [8,6]$ → $\langle 7,8,8,6\rangle$

10 $[9,10] \times \begin{bmatrix}12,8\\12,9\end{bmatrix}$ → $\emptyset$

$[11,12] \times \begin{bmatrix}12,8\\12,9\end{bmatrix}$ → $\langle 11,12,12,8\rangle$ $\langle 11,12,12,9\rangle$

EHJ $h(k) = k \% 4$

NLJ or any other algo in memory

$h(B)$

R: 0 → $\langle 3,4\rangle$ $\langle 7,8\rangle$ $\langle 11,12\rangle$ | 1 → | 2 → $\langle 1,2\rangle$ $\langle 5,6\rangle$ $\langle 9,10\rangle$ | 3 →

S: 0 → $\langle 12,8\rangle$ $\langle 12,9\rangle$ $\langle 4,4\rangle$ $\langle 8,6\rangle$ | 1 → | 2 → $\langle 2,2\rangle$ $\langle 2,7\rangle$ $\langle 6,2\rangle$ | 3 →

$R_0 \bowtie S_0 \Rightarrow \langle 3,4,4,4\rangle$ $\langle 7,8,8,6\rangle$ $\langle 11,12,12,8\rangle$ $\langle 11,12,12,9\rangle$

$R_1 \bowtie S_1 \Rightarrow \emptyset$

$R_2 \bowtie S_2 \Rightarrow \langle 1,2,2,2\rangle$ $\langle 1,2,2,7\rangle$ $\langle 5,6,6,2\rangle$

$R_3 \bowtie S_3 \Rightarrow \emptyset$

only join shared buckets

```
SELECT
  lineitem.orderkey,
  sum(lineitem.extendedprice*(1-lineitem.discount)) as
revenue,
  orders.orderdate,
  orders.shippriority
FROM
  customer,
  orders,
  lineitem
WHERE
  customer.mktsegment = 'BUILDING' and customer.custkey =
orders.custkey
  and lineitem.orderkey = orders.orderkey
  and orders.orderdate < DATE('1995-03-15')
  and lineitem.shipdate > DATE('1995-03-15')
GROUP BY lineitem.orderkey, orders.orderdate,
orders.shippriority
ORDER BY revenue desc, orders.orderdate;
```

Optimization