Database Administration

José Orlando Pereira

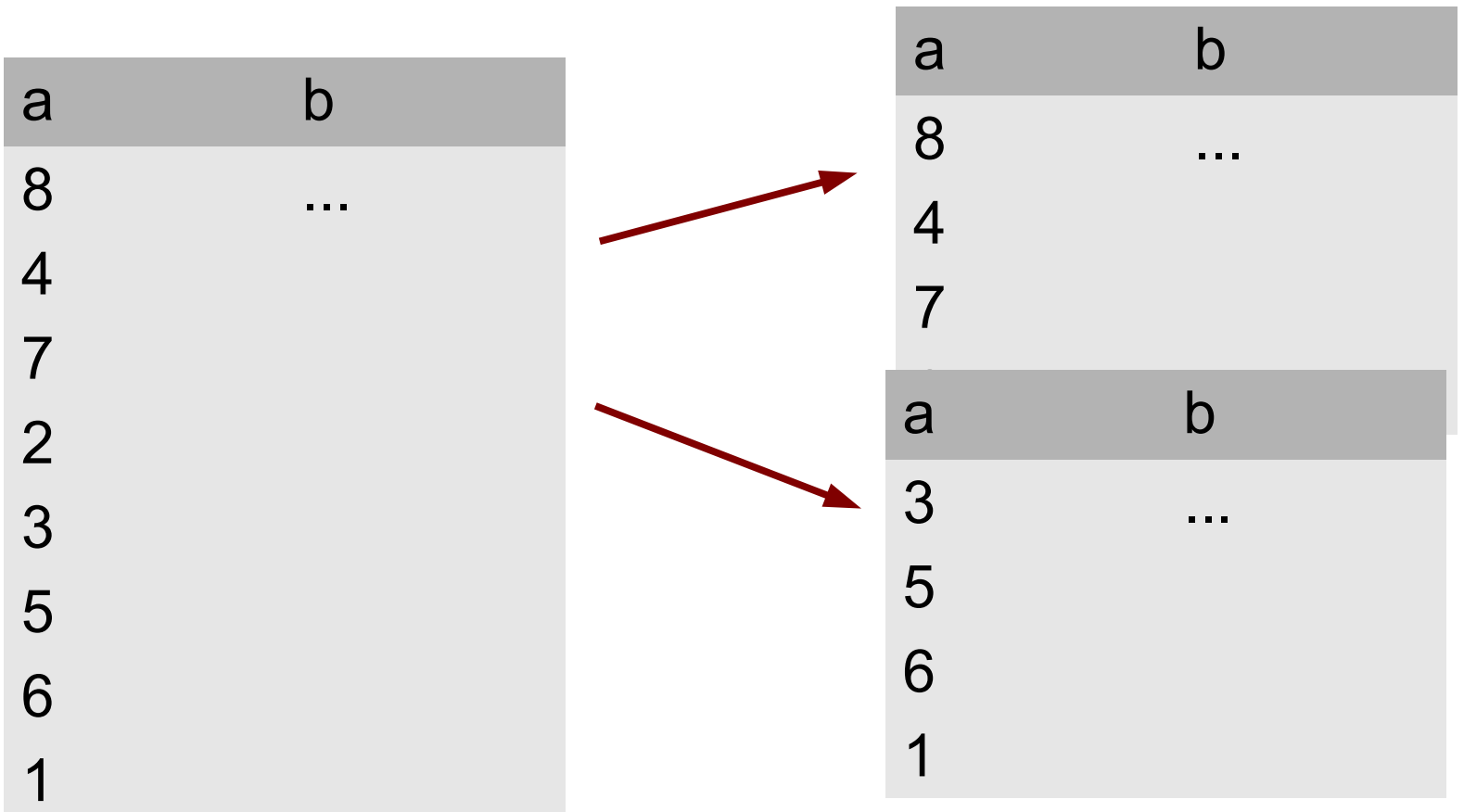
Departamento de Informática
Universidade do Minho

Large relations and sorting

- Algorithms using sorted data are more efficient (e.g. than nested loops)
- How to sort data that does not fit in memory?

Merge-sort

- Split data in chunks that fit in memory:



Merge-sort

- Load and sort each of them:

a	b
2	...
4	
7	
8	

a	b
3	...
5	
6	
1	

Merge-sort

- Load and sort each of them:

a	b
2	...
4	
7	
8	

a	b
1	...
3	
5	
6	

Merge-sort

- When iterating, select the next record from the fragment with the next key:

a	b
2	...
4	
7	
8	

a	b
1	...
3	
5	
6	

Merge-sort

- When iterating, select the next record from the fragment with the next key:

a	b
2	...
4	
7	
8	

a	b
1	...
3	
5	
6	

Merge-sort

- When iterating, select the next record from the fragment with the next key:

a	b
2	...
4	
7	
8	

a	b
1	...
3	
5	
6	

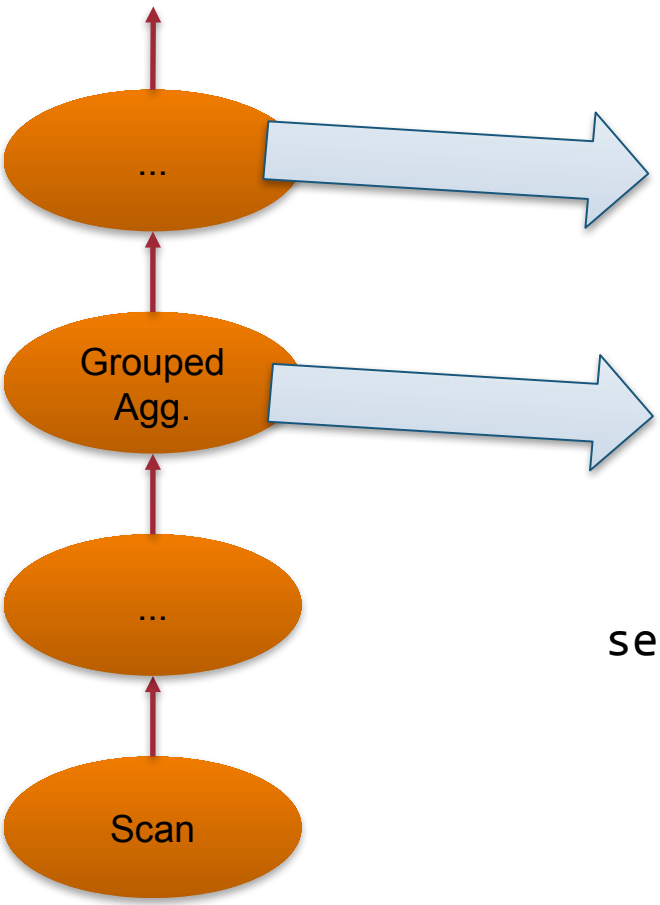
Two-pass, full relation, unary

- First pass is sorting
- Duplicate elimination:
  - Cache last record
  - “select distinct * from X;”
- Grouping and aggregation:
  - Cache last group
  - “select count(*) from X group by b;”

Two-pass, full relation, binary

- Union, difference, intersection, product, join:
  - Read record R1 from sorted relation T1
  - Read record R2 from sorted relation T2
  - If R1 = R2:
    - Use R1 and R2
  - If R1 < R2:
    - R1 does not exist in T2
    - Skip R1
  - If R2 < R1
    - ....

Parallel execution

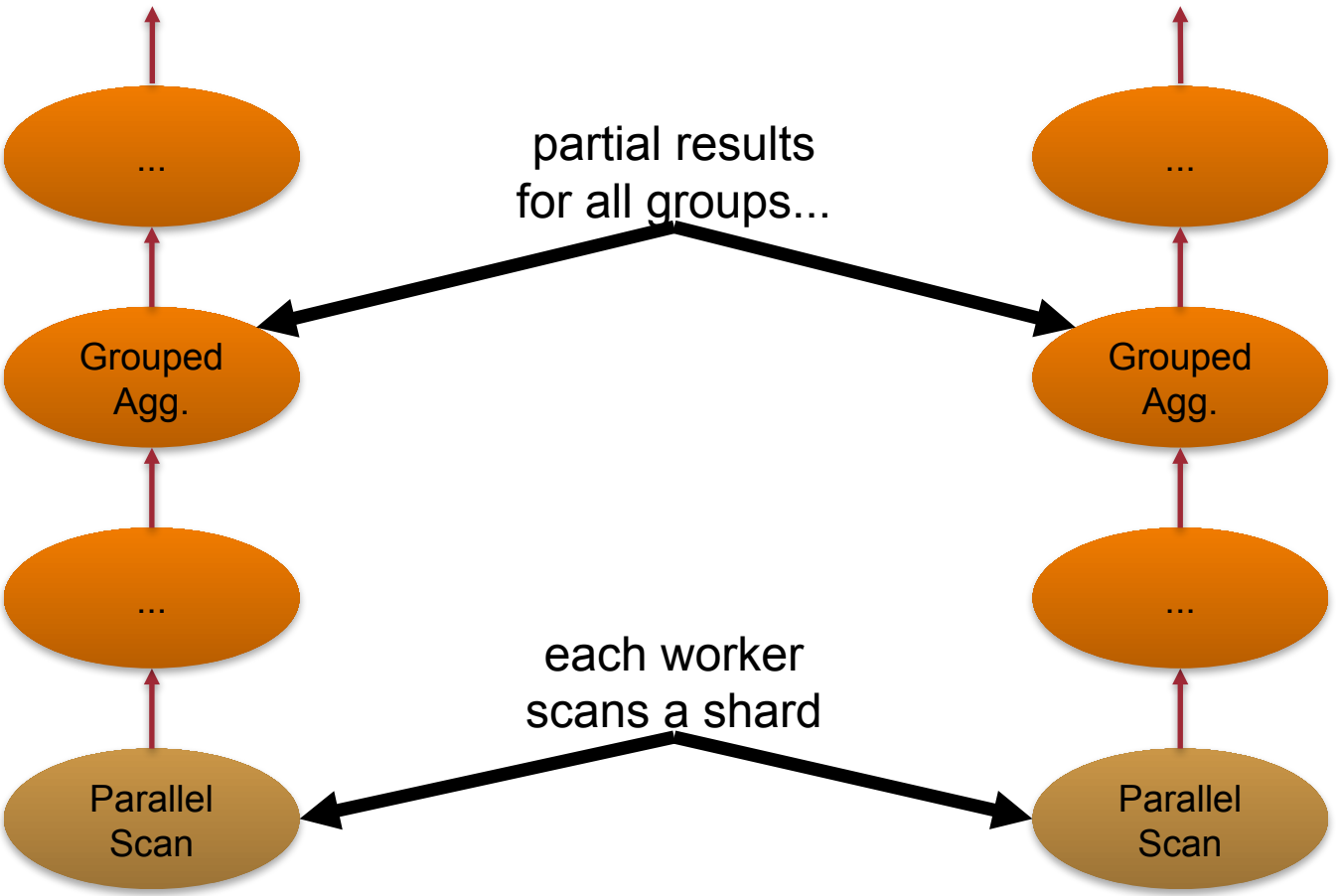


stateless operator

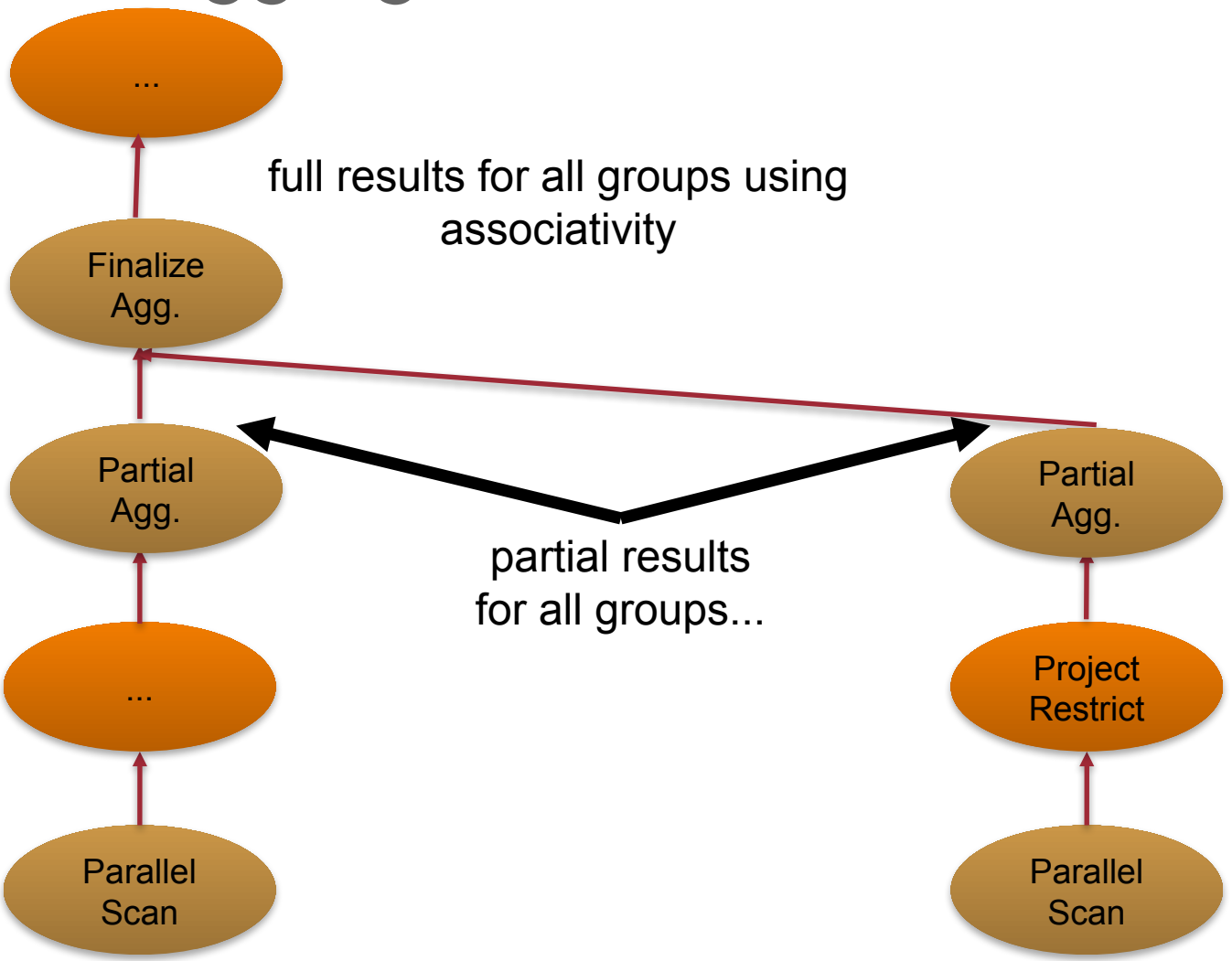
stateful operator

```
select a, avg(b) from t
    where c > 0
    group by a
    having avg(b) > 0
```

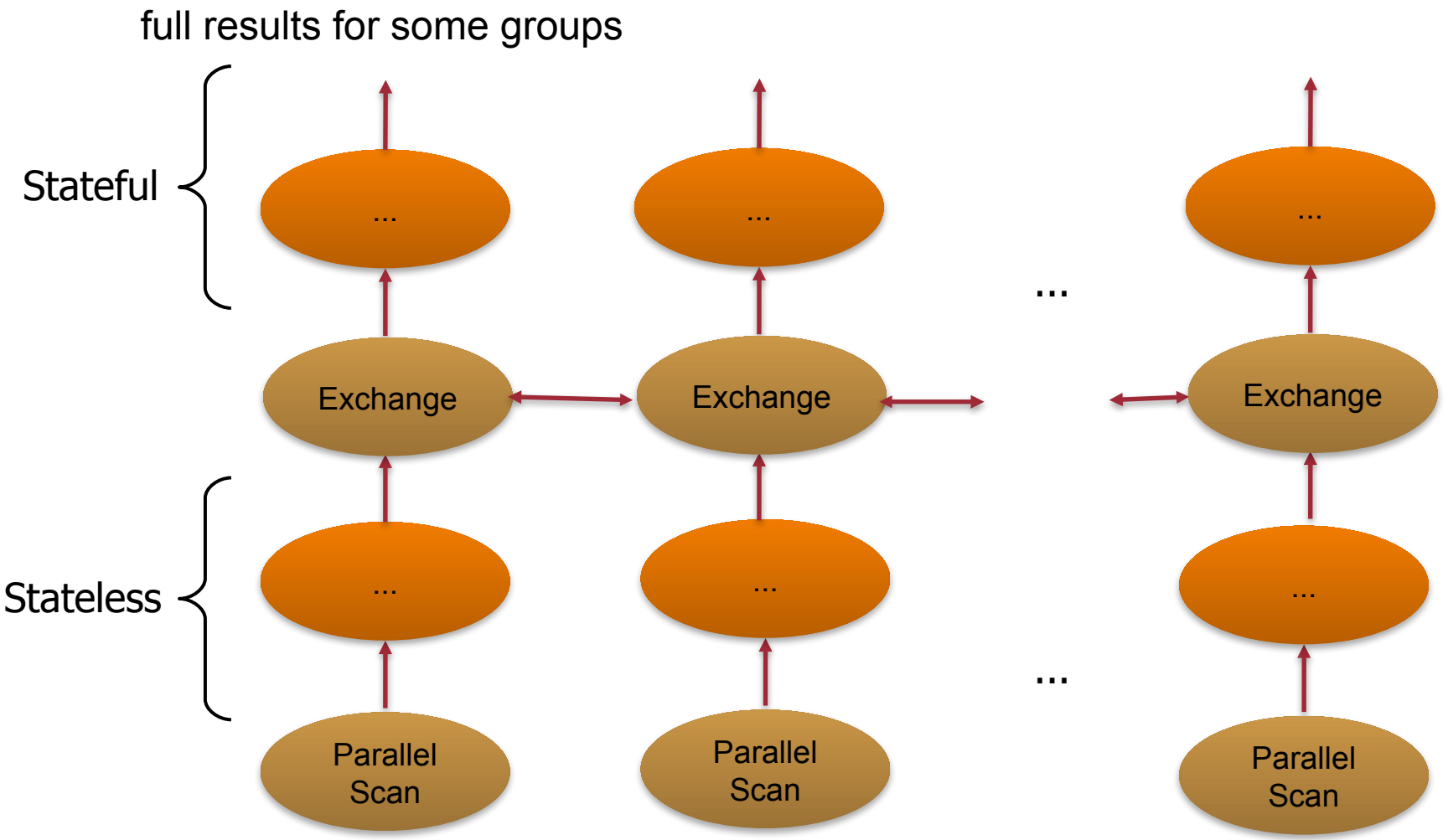
Parallel execution



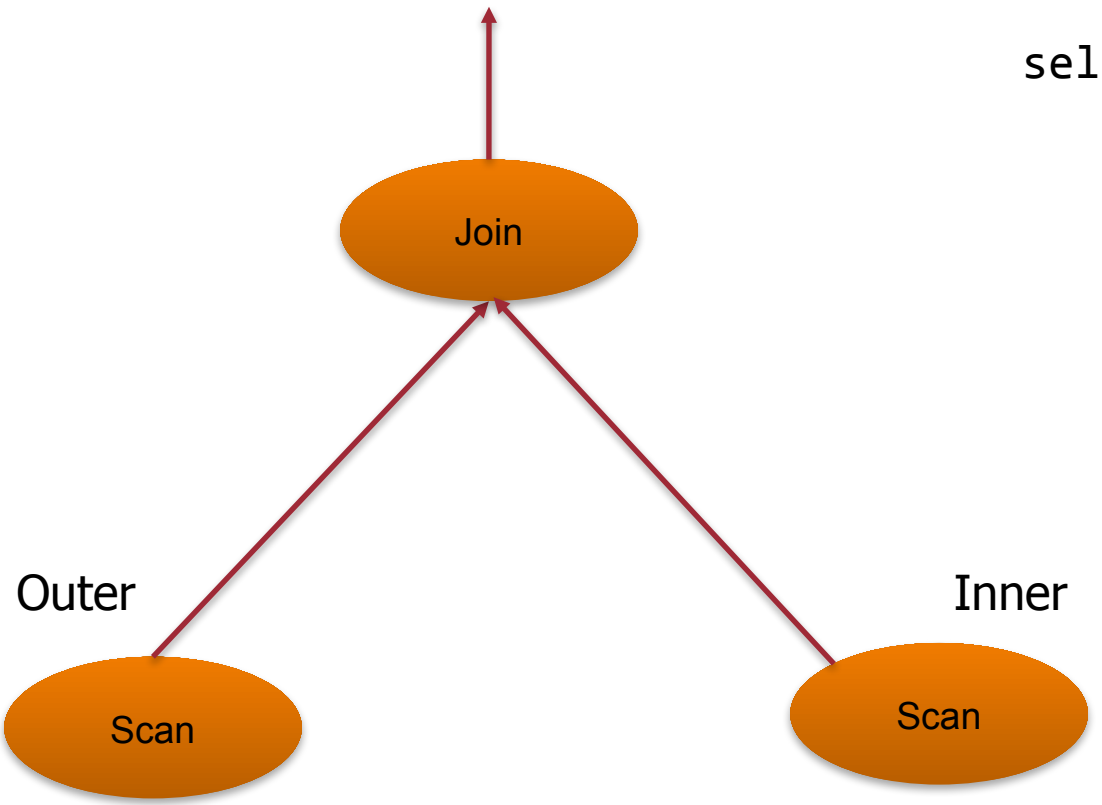
Example: Aggregation



General approach: Exchange operator

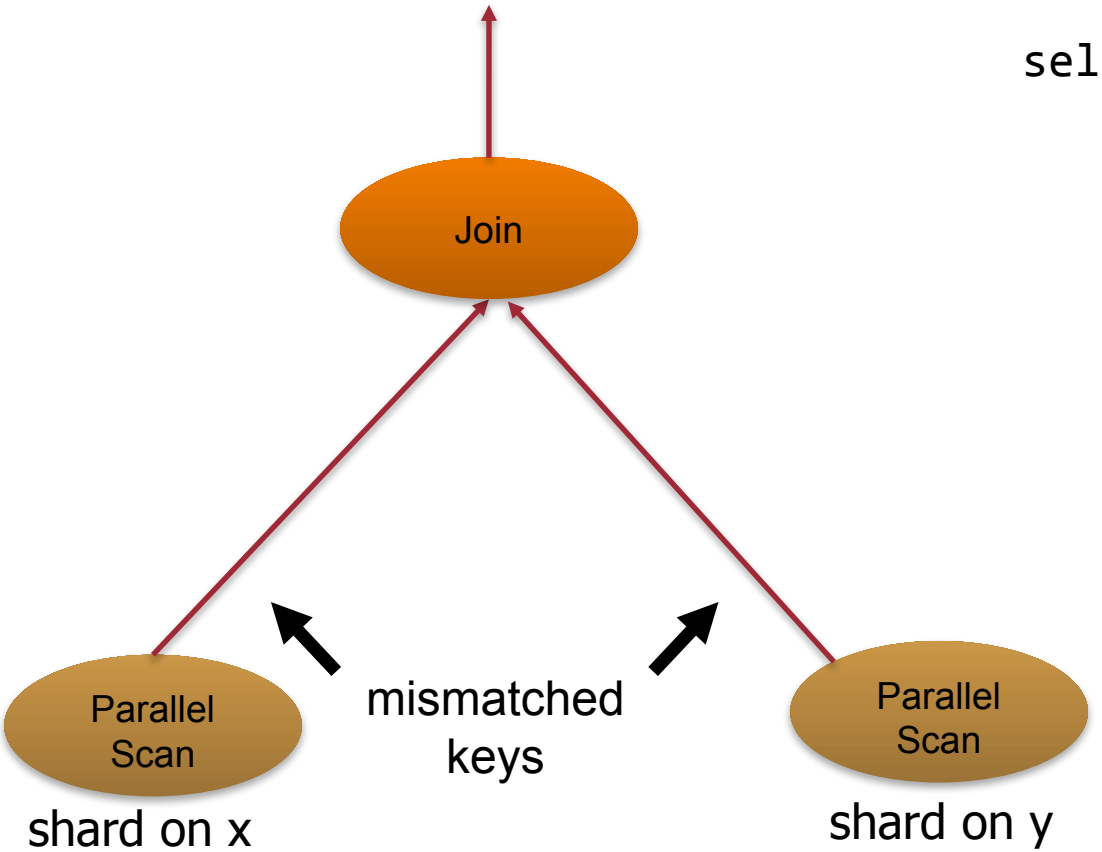


Parallel execution: Join



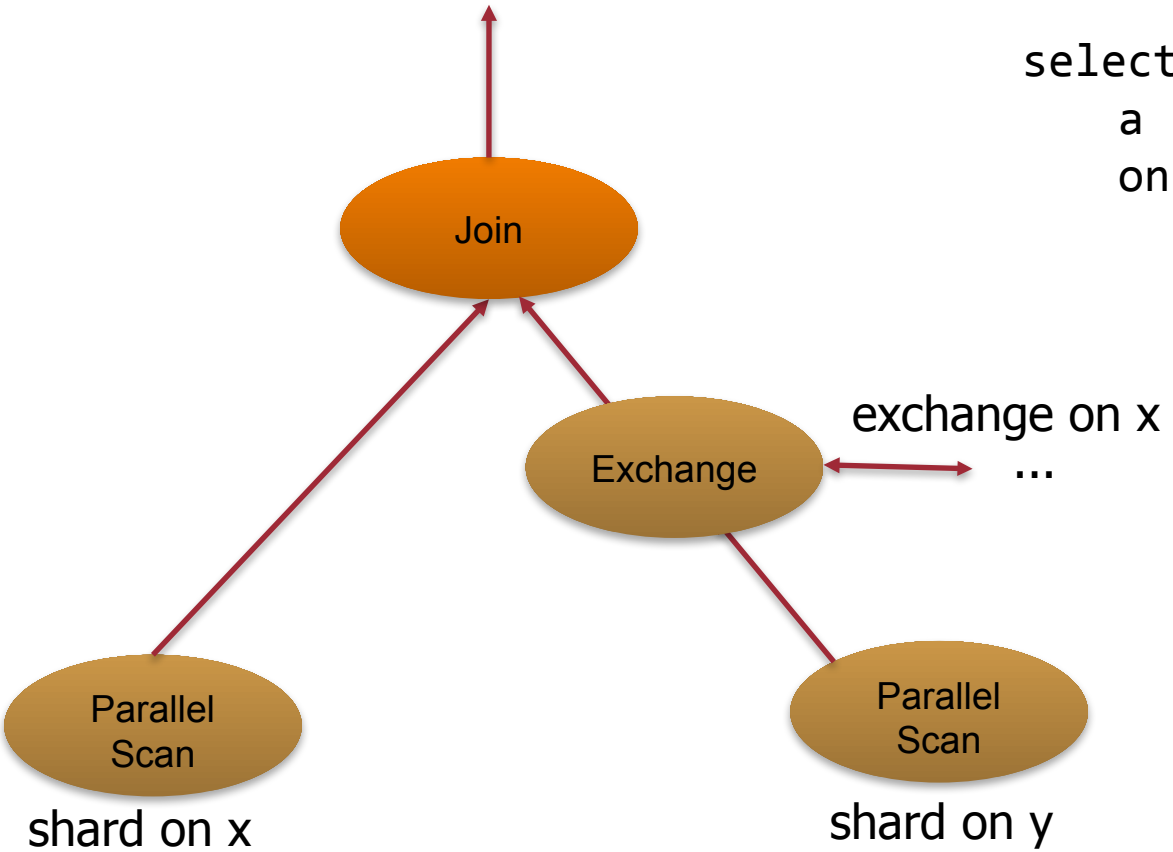

```
select ... from
    a inner join b
    on a.x = b.x
```

Parallel execution: Join



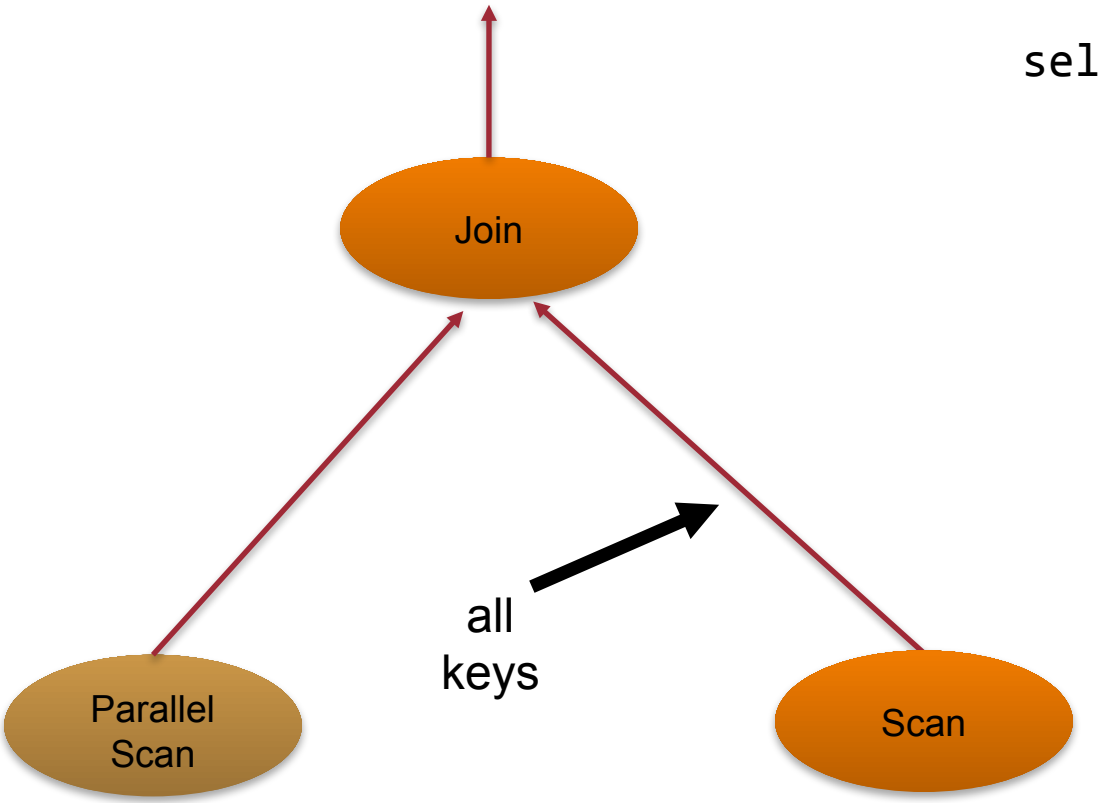

```
select ... from
    a inner join b
    on a.x = b.x
```

Parallel execution: Join



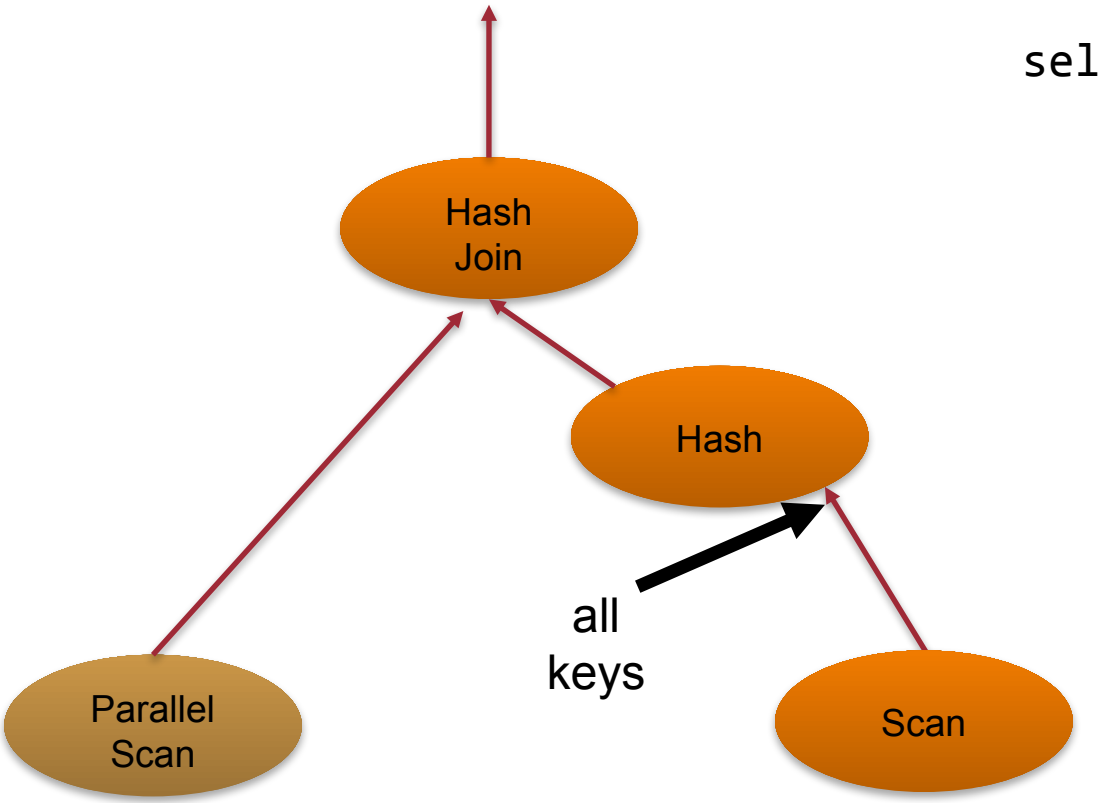

```
select ... from
    a inner join b
    on a.x = b.x
```

Parallel execution: Join




```
select ... from
    a inner join b
    on a.x = b.x
```

Parallel execution: Join




```
select ... from
    a inner join b
    on a.x = b.x
```

Summary

- Needed for multi-core execution
- Alternatives:
  - Sub-plan to execute in parallel
  - Alternatives for stateful operators
  - Number of workers

Conclusion

- There are a number of options for executing each query
- More options if we consider other data structures
- Varying performance:
  - Memory requirements
  - Number of iterations
  - Disk accesses
- What is the best one?
- How can it be discovered?