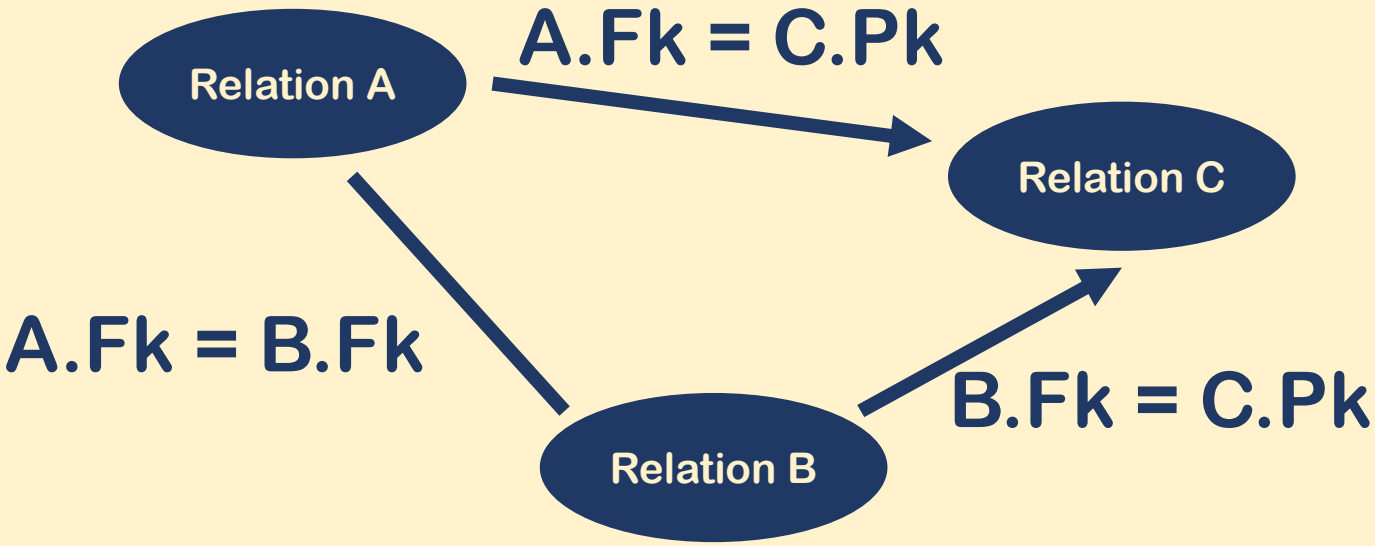
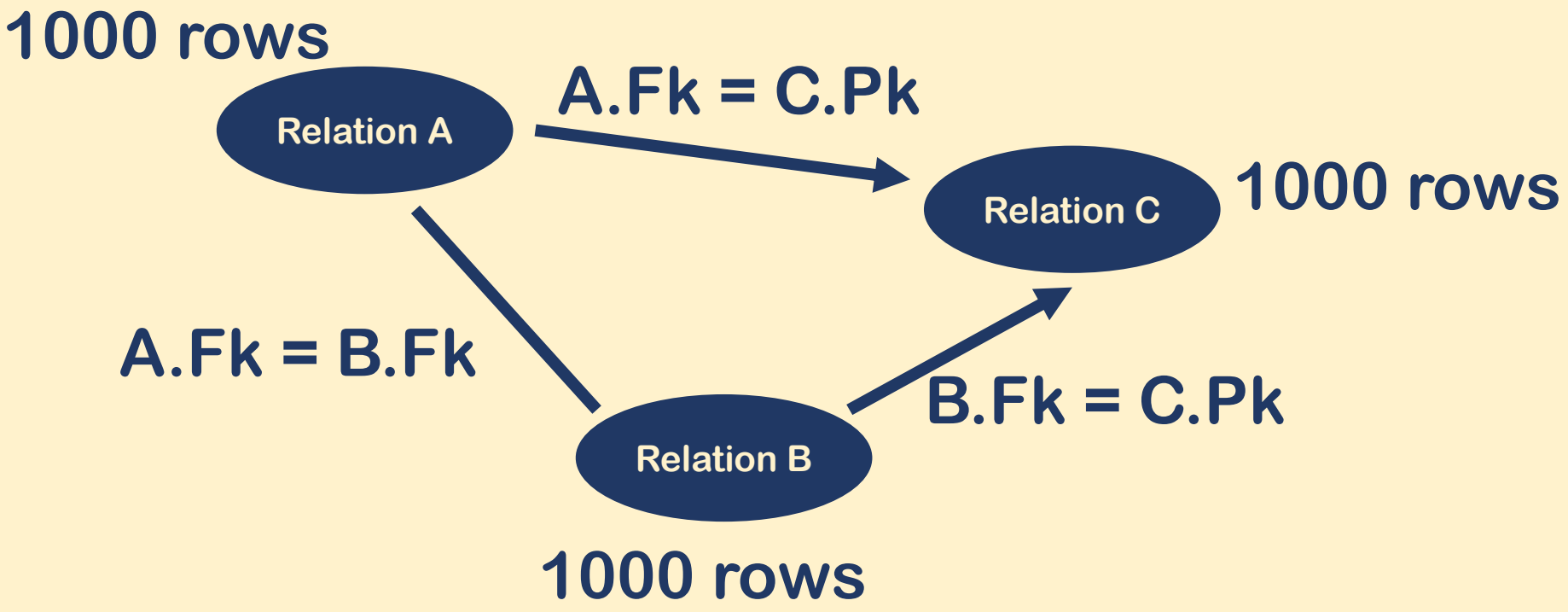
Efficient Query Re-optimization with Judicious Subquery Selections

Junyi Zhao, Huanchen Zhang, Yihan Gao
Tsinghua University
SIGMOD 2023

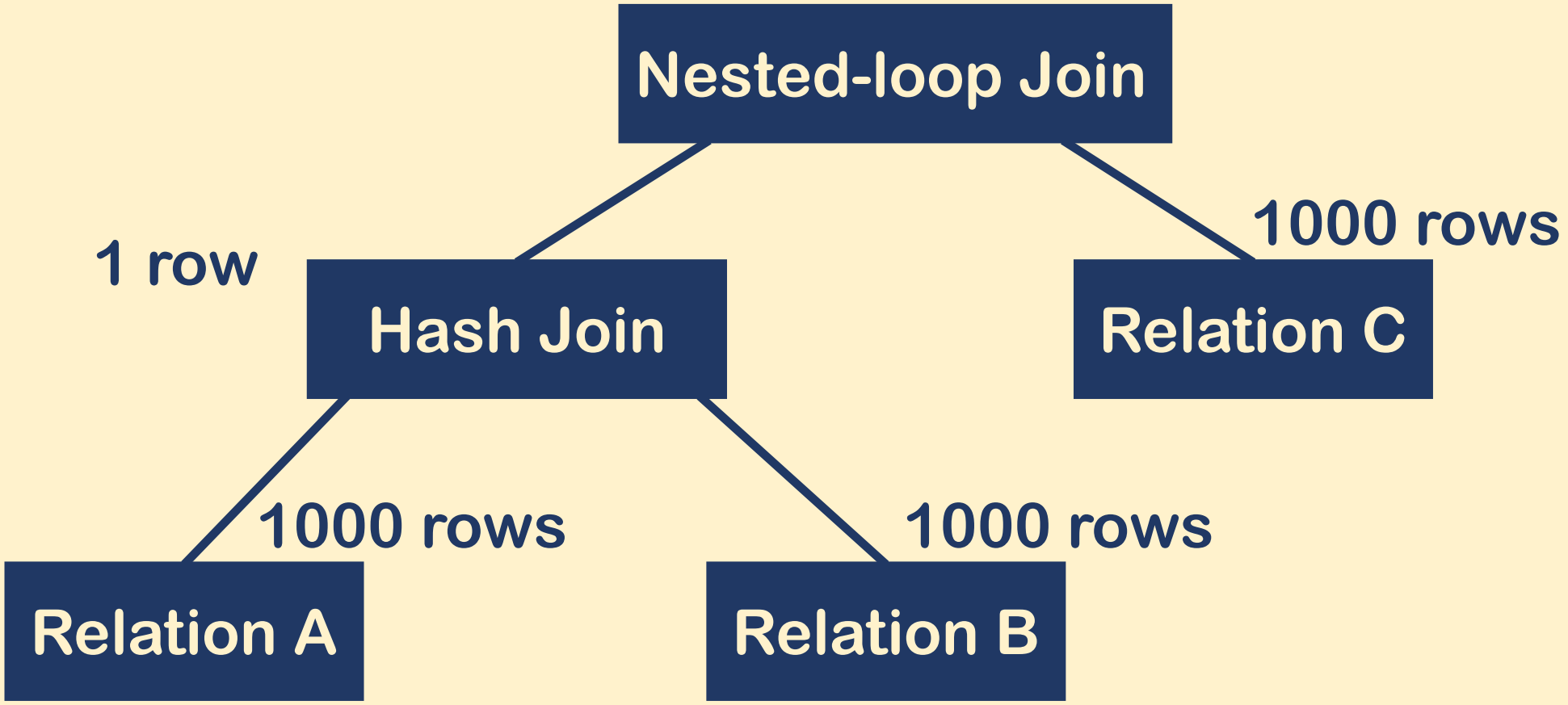
Wrong CE leads to bad physical plan



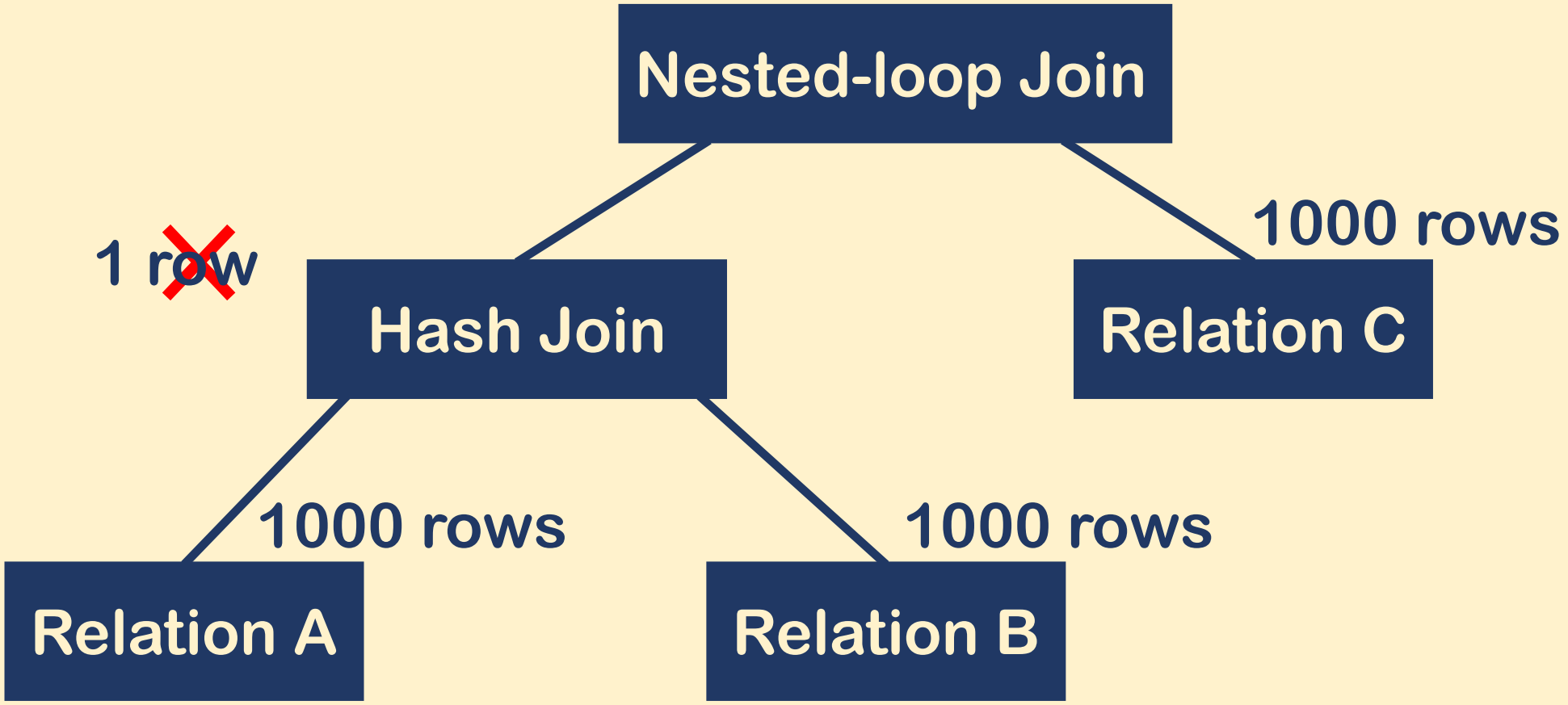
Wrong CE leads to bad physical plan



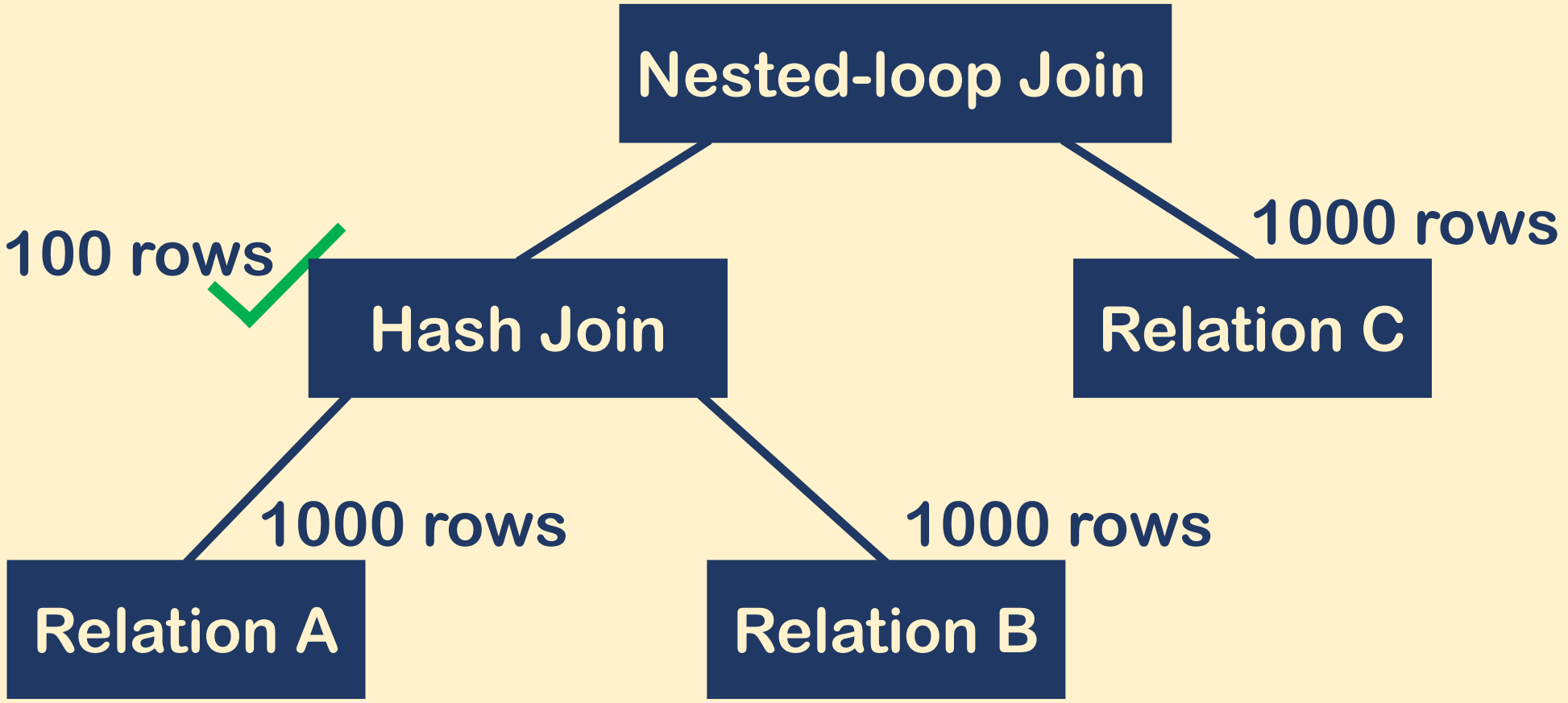
Wrong CE leads to bad physical plan



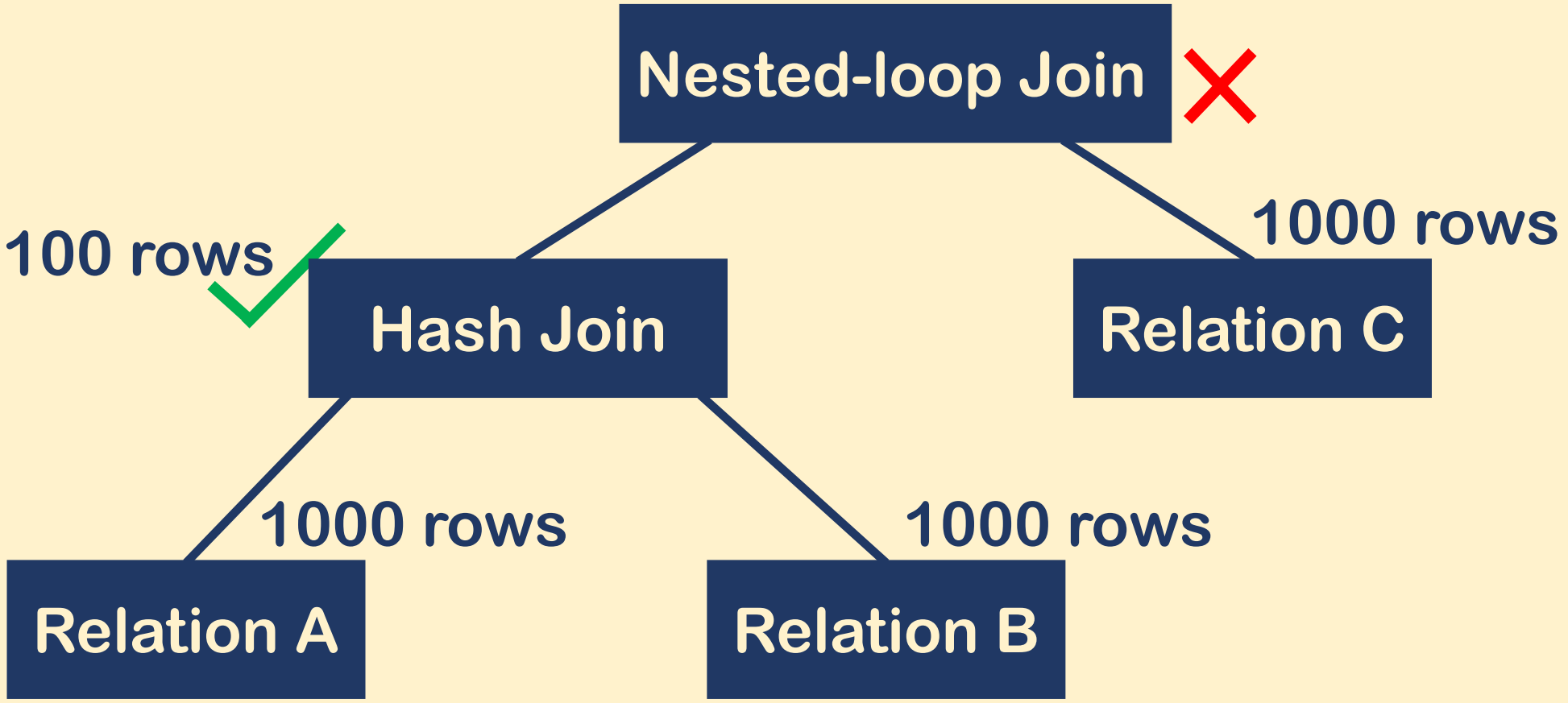
Wrong CE leads to bad physical plan



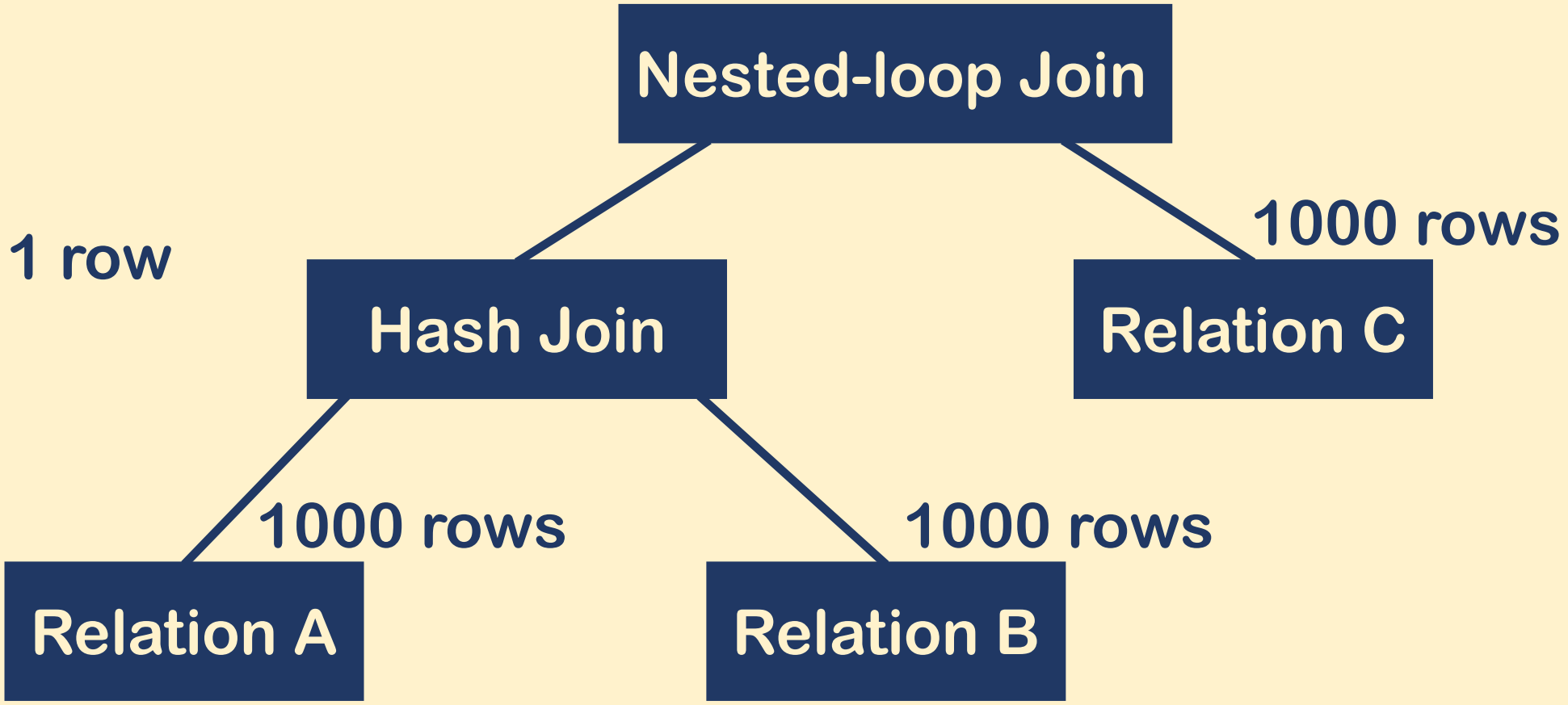
Wrong CE leads to bad physical plan



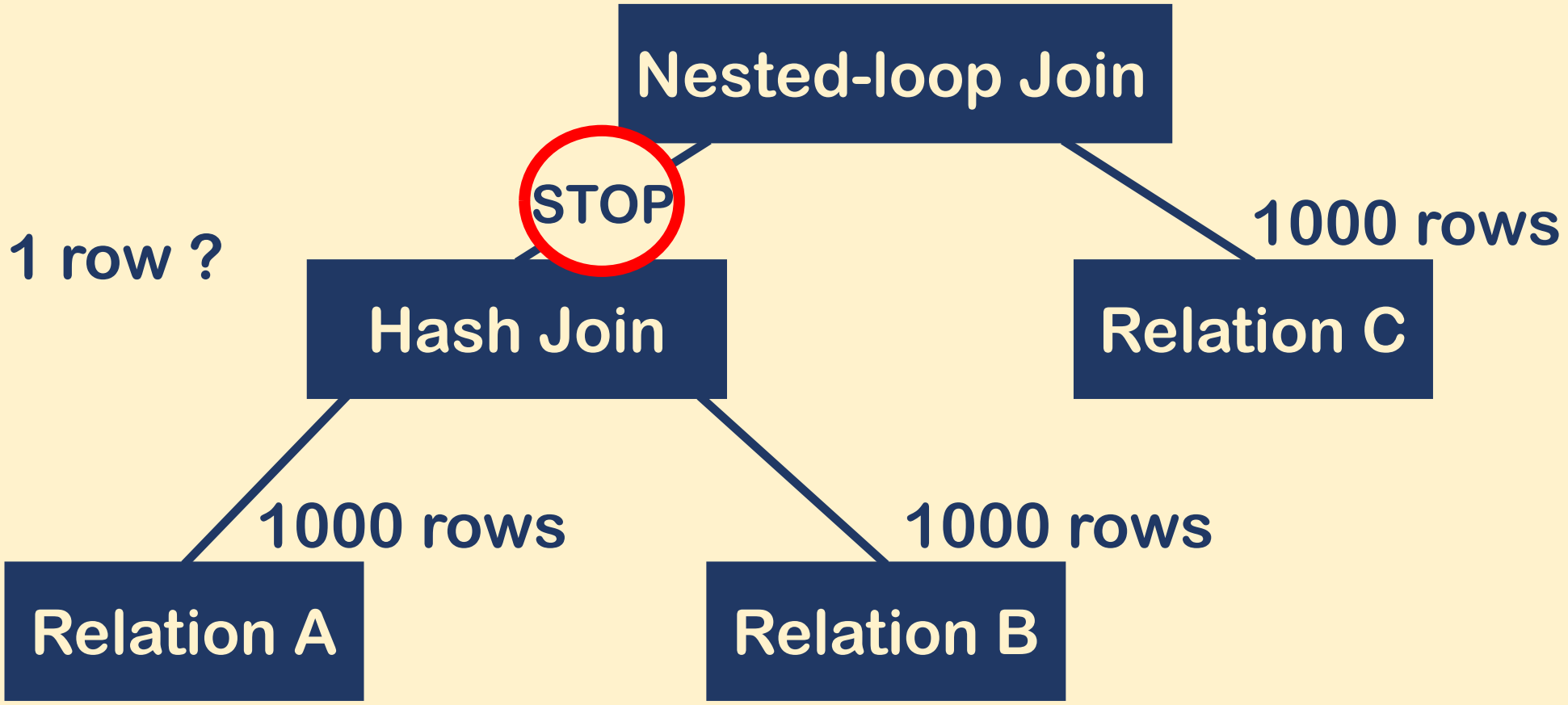
Wrong CE leads to bad physical plan



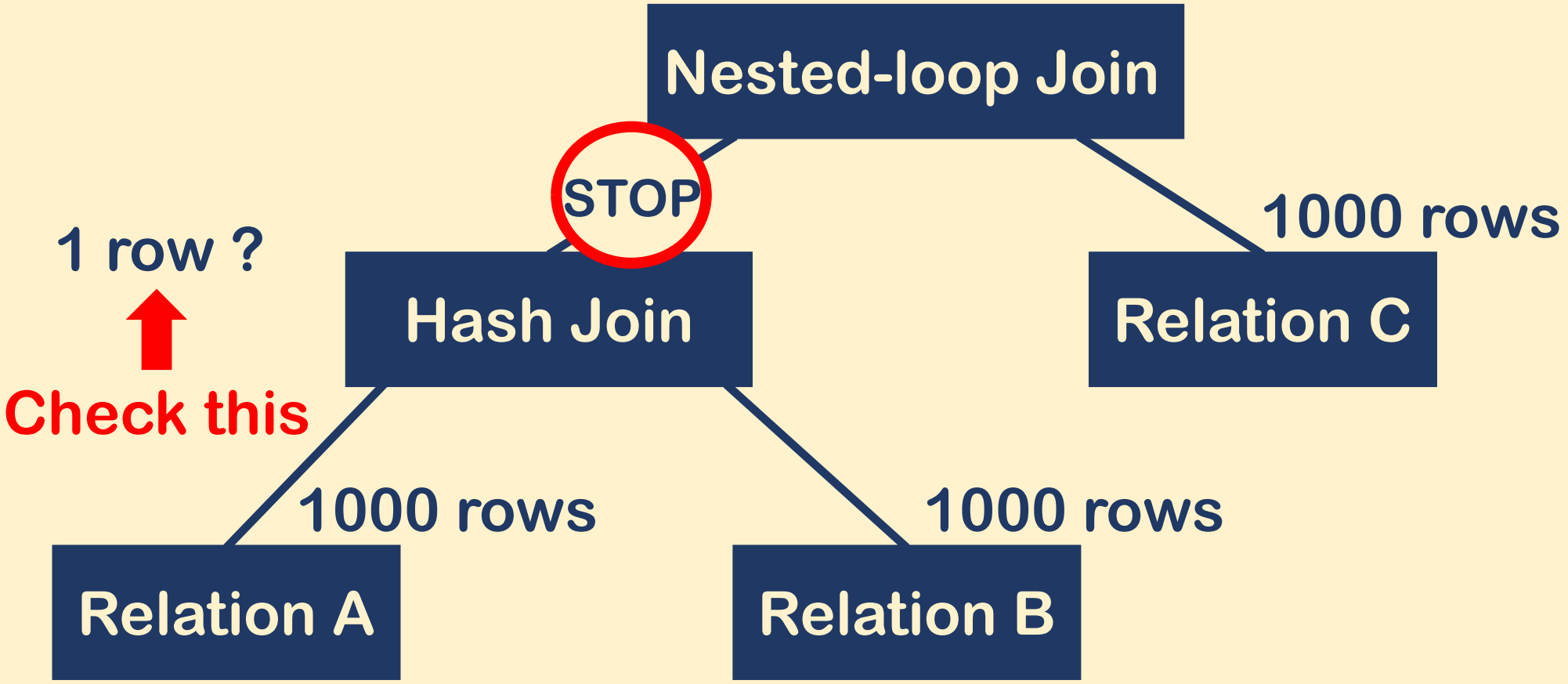
Re-optimization fixes mistakes



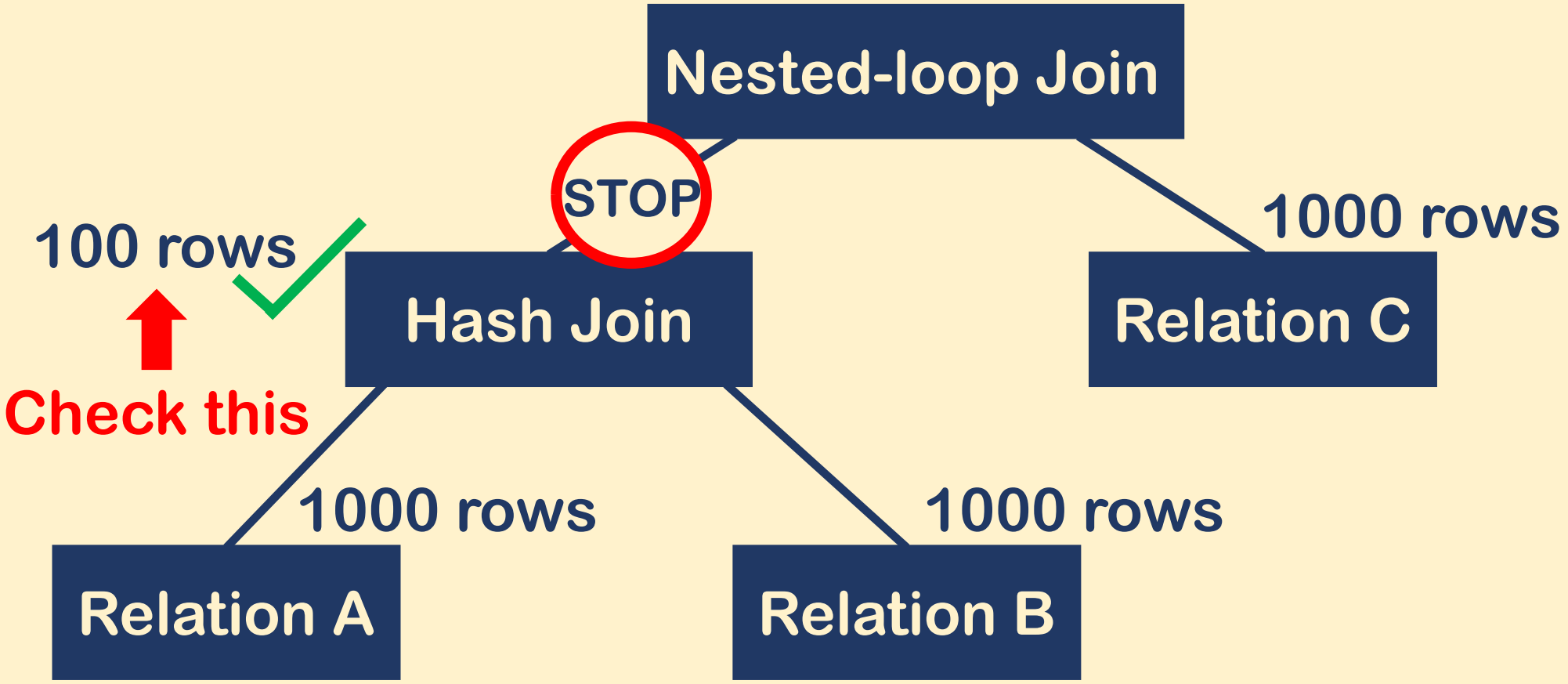
Re-optimization fixes mistakes



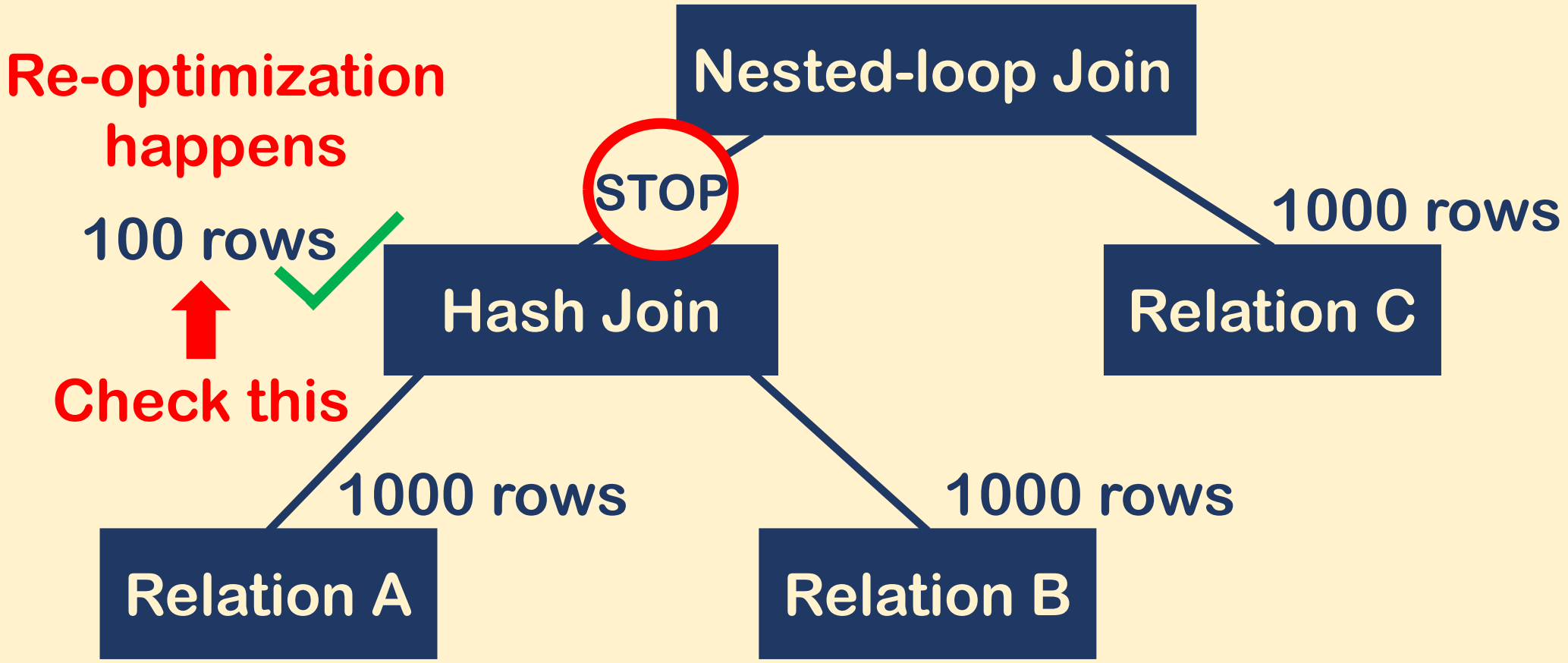
Re-optimization fixes mistakes



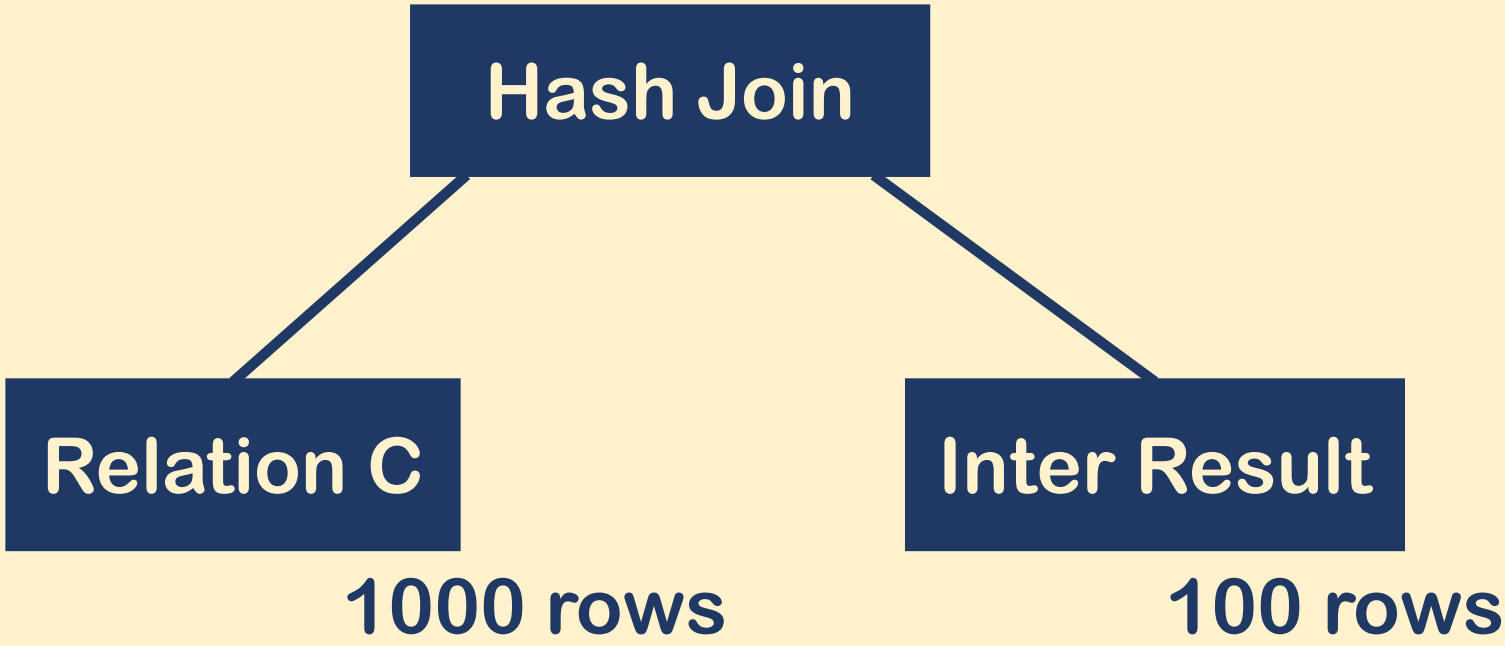
Re-optimization fixes mistakes



Re-optimization fixes mistakes

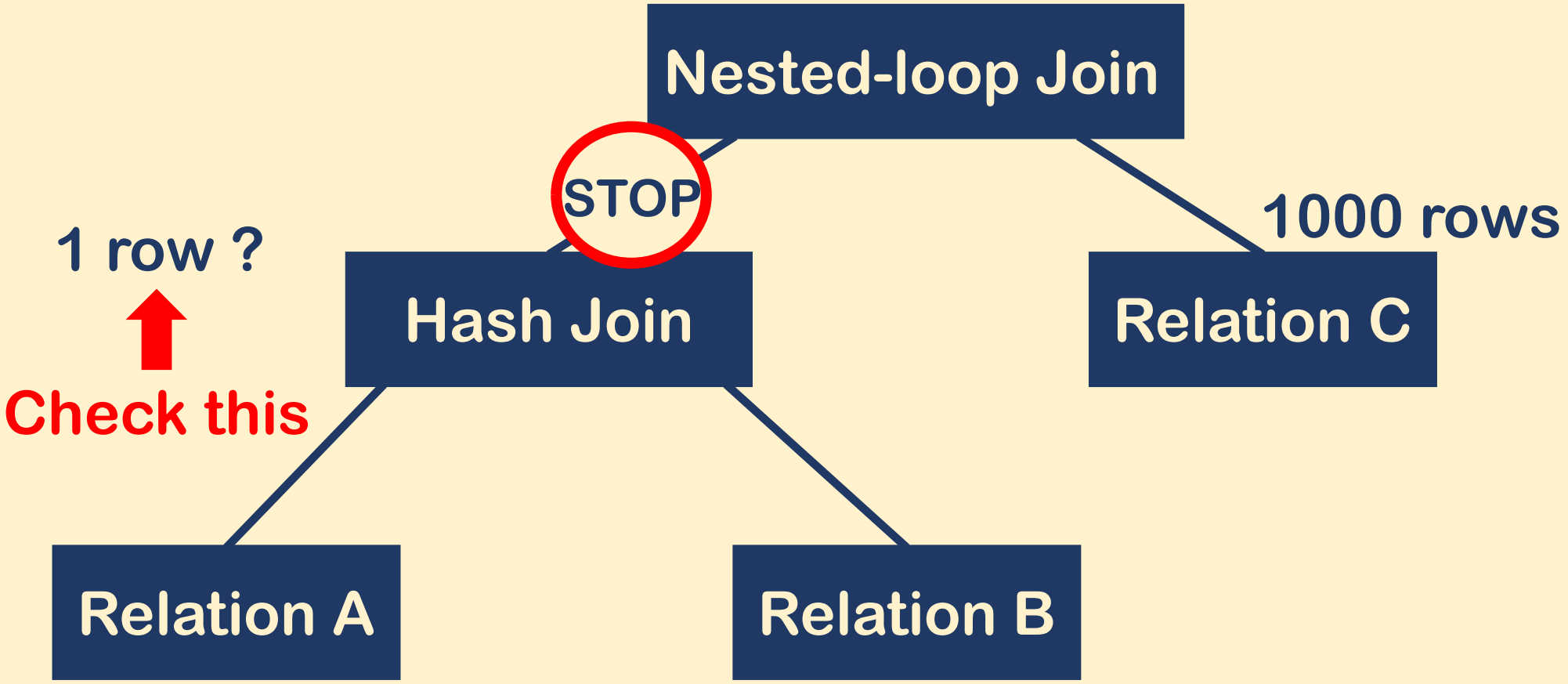


Re-optimization fixes mistakes

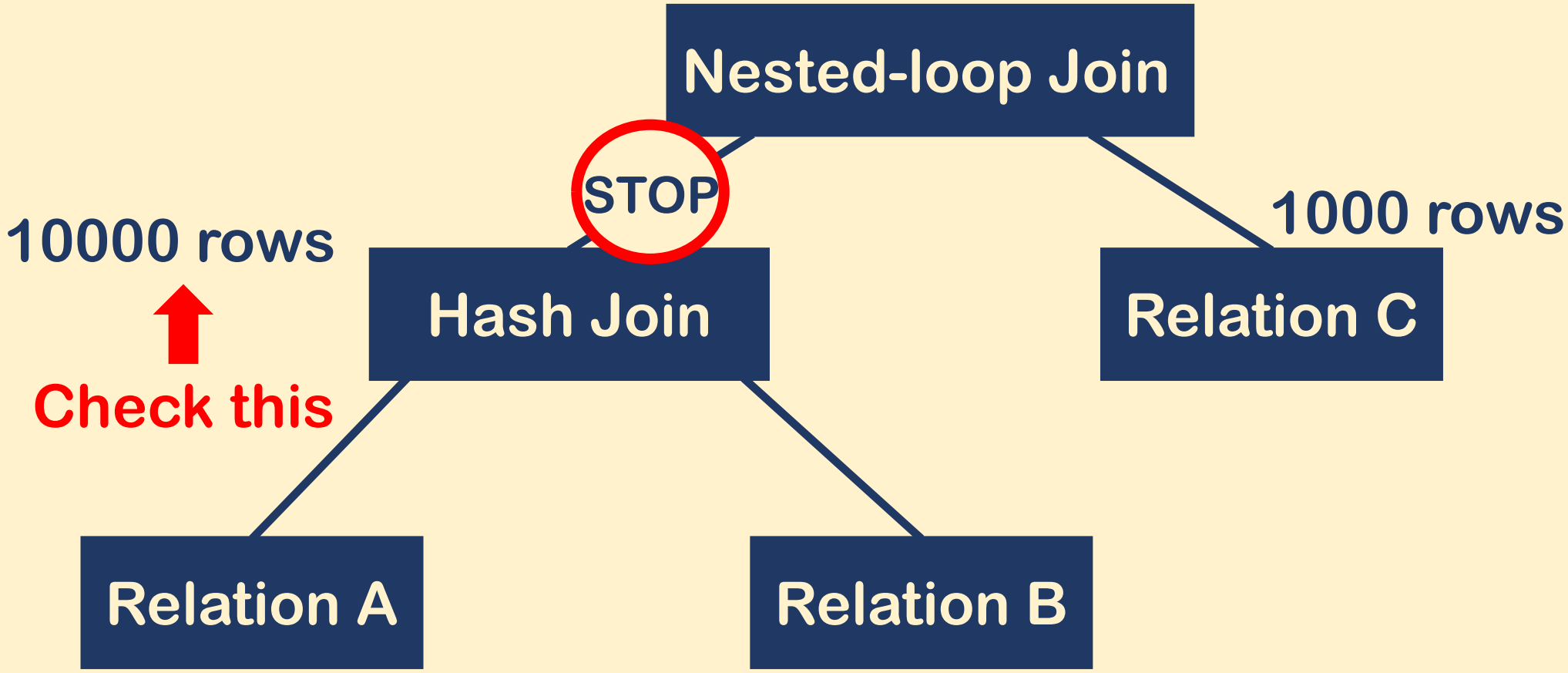


Mistake is too large to fix

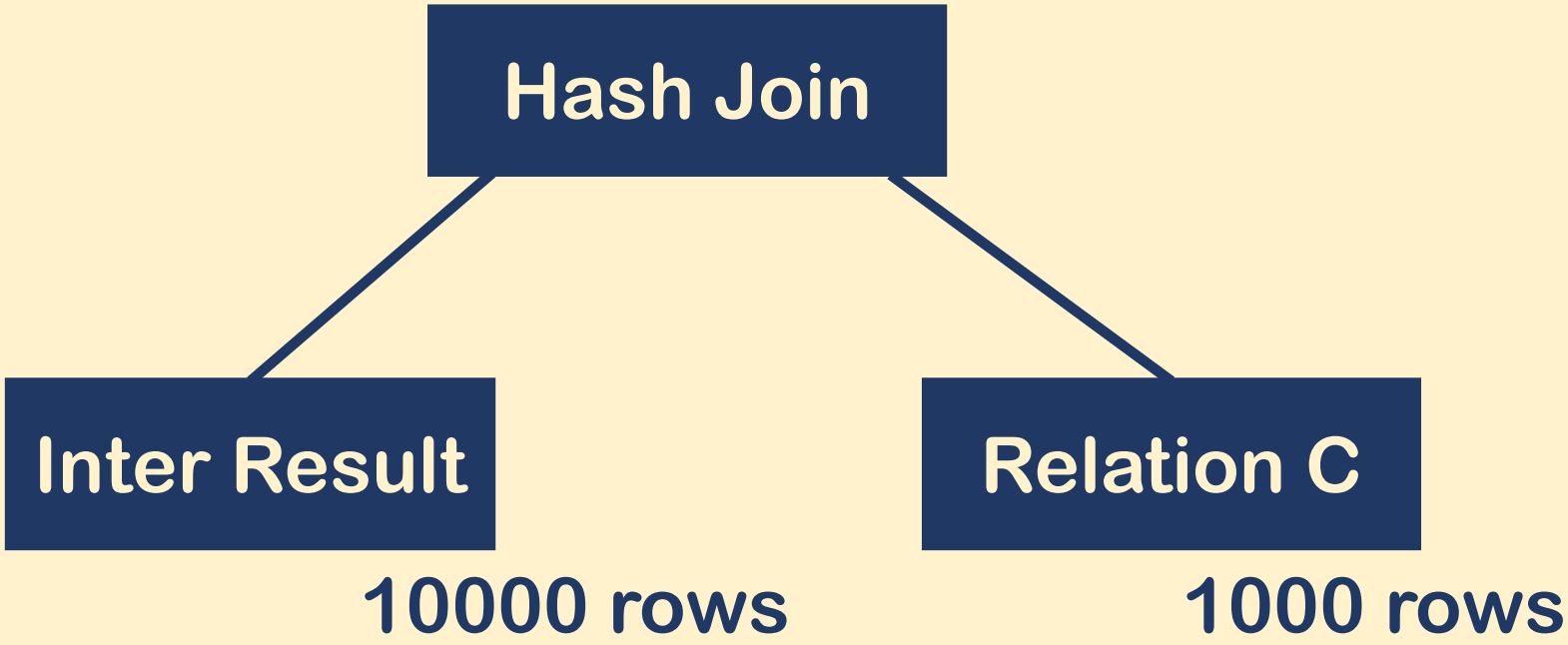
Mistake is too large to fix



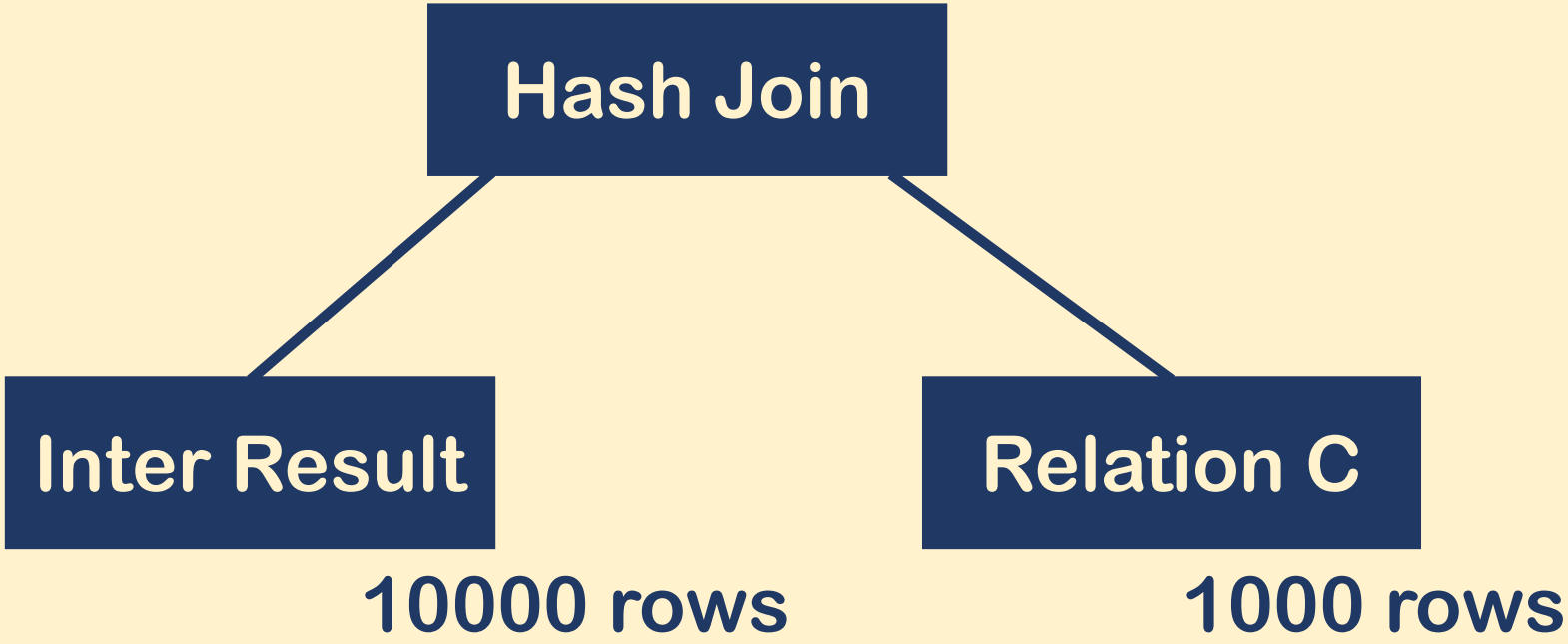
Mistake is too large to fix



Mistake is too large to fix

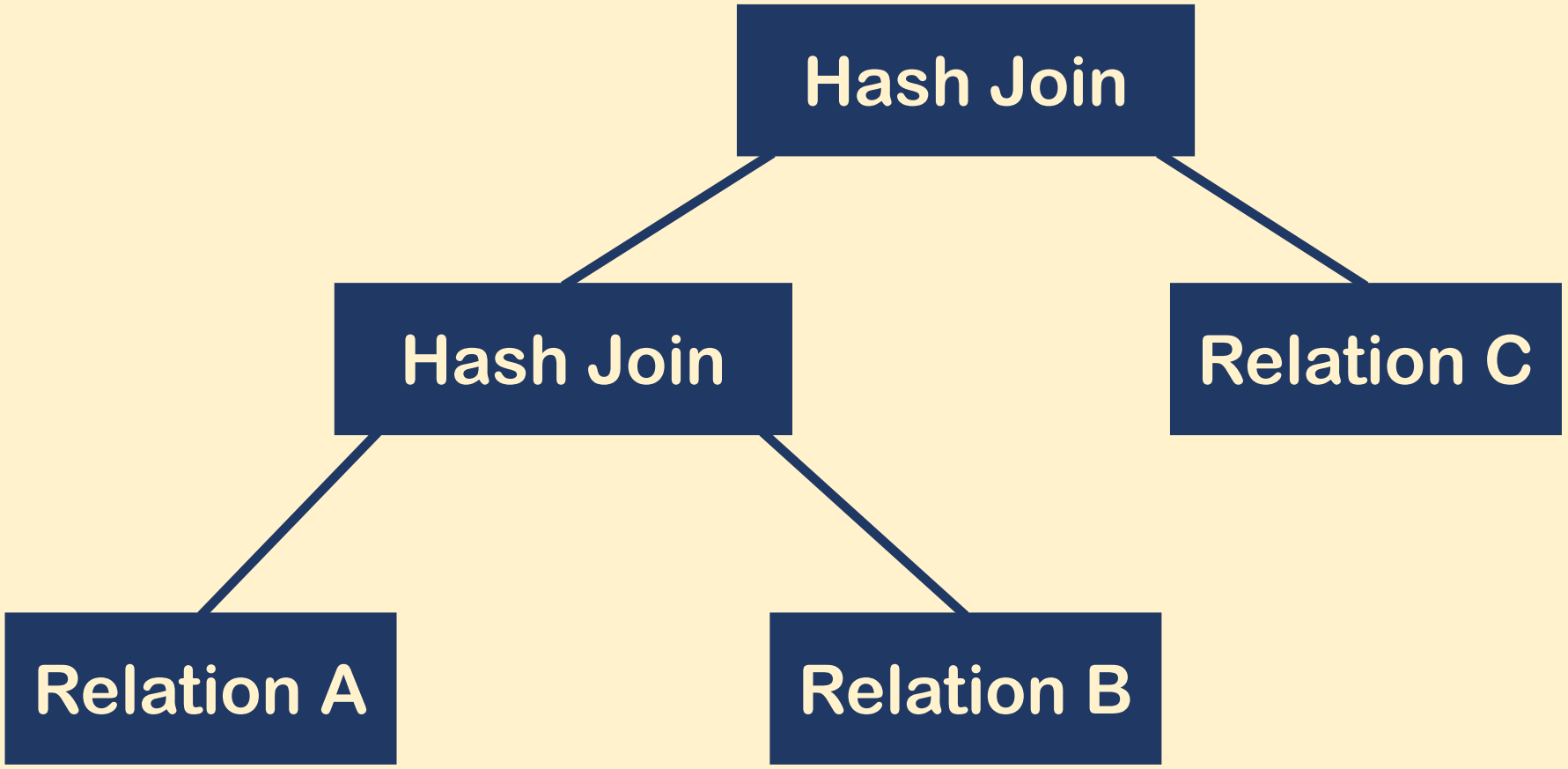


Mistake is too large to fix

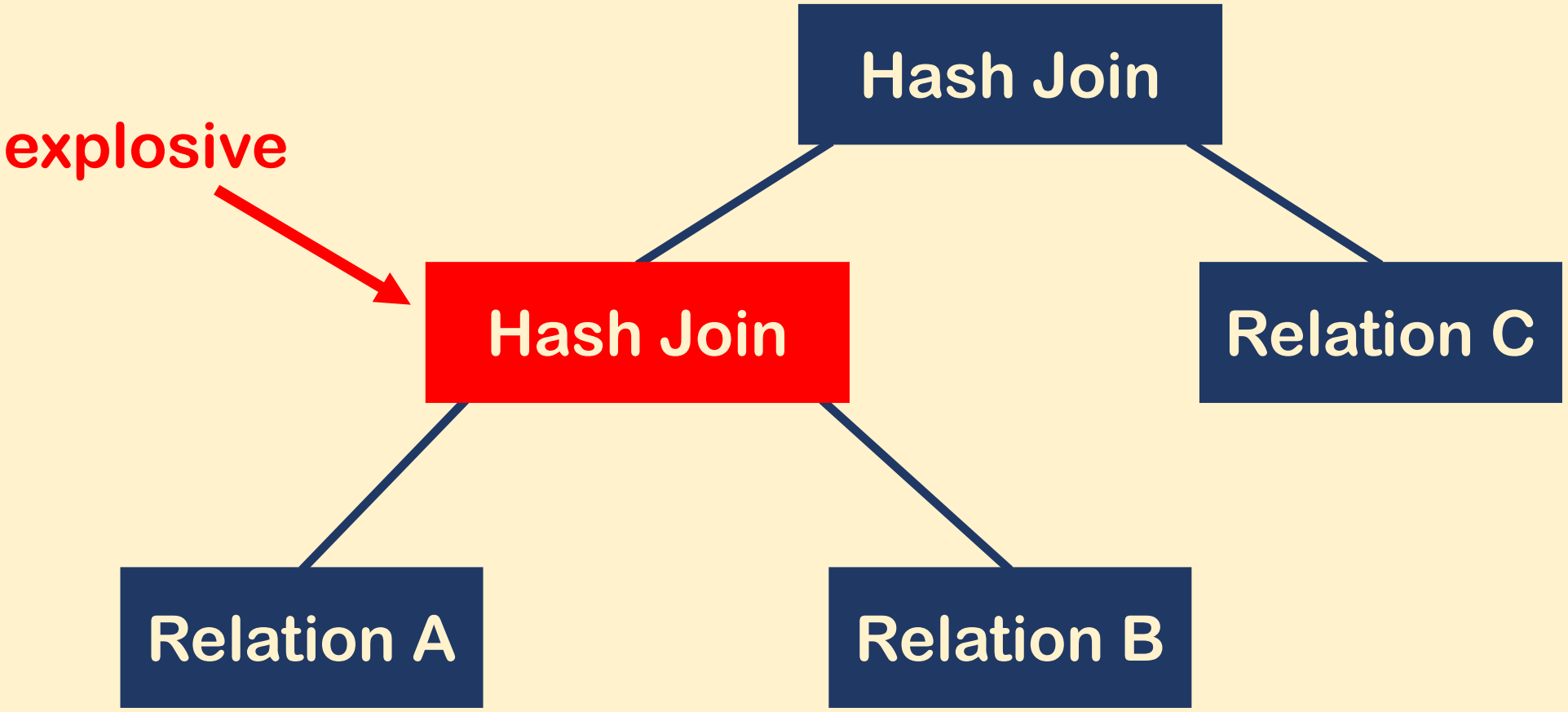


Re-optimization:
worked but useless

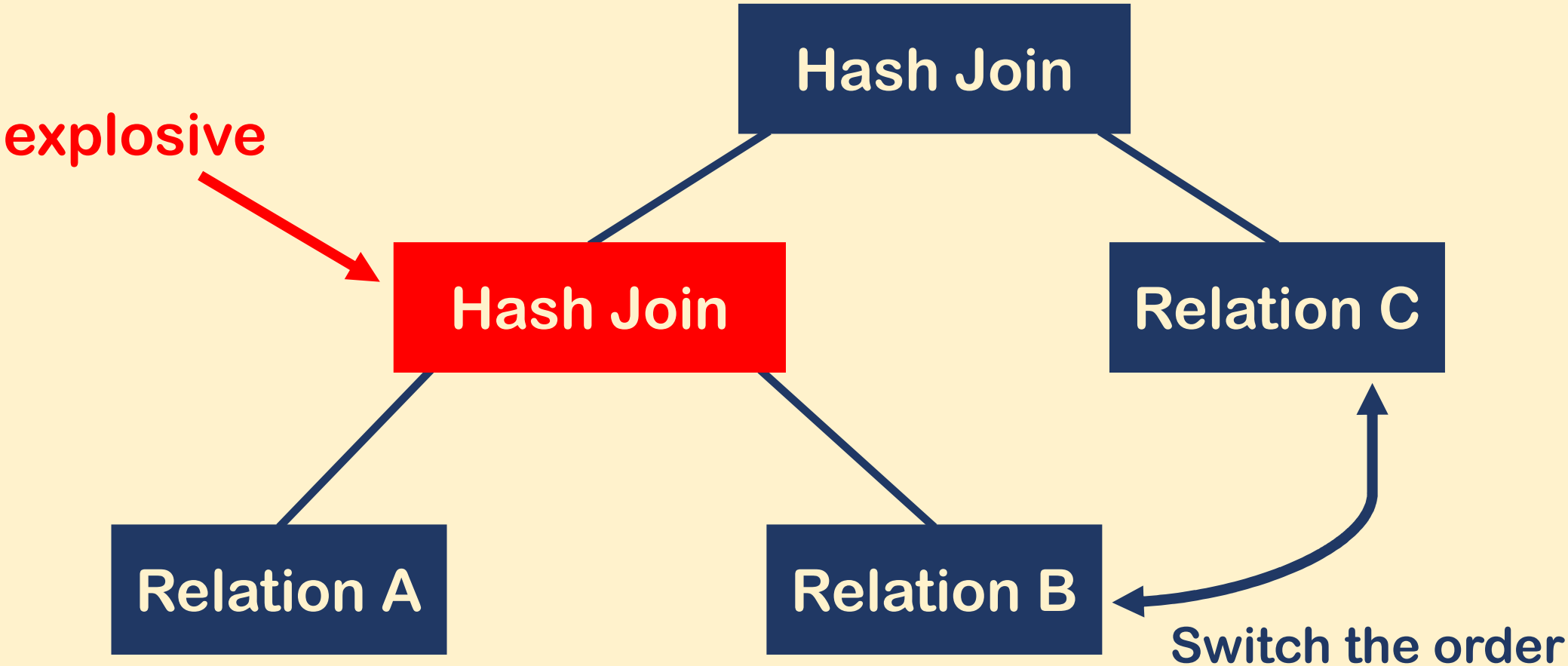
Avoid such explosive join



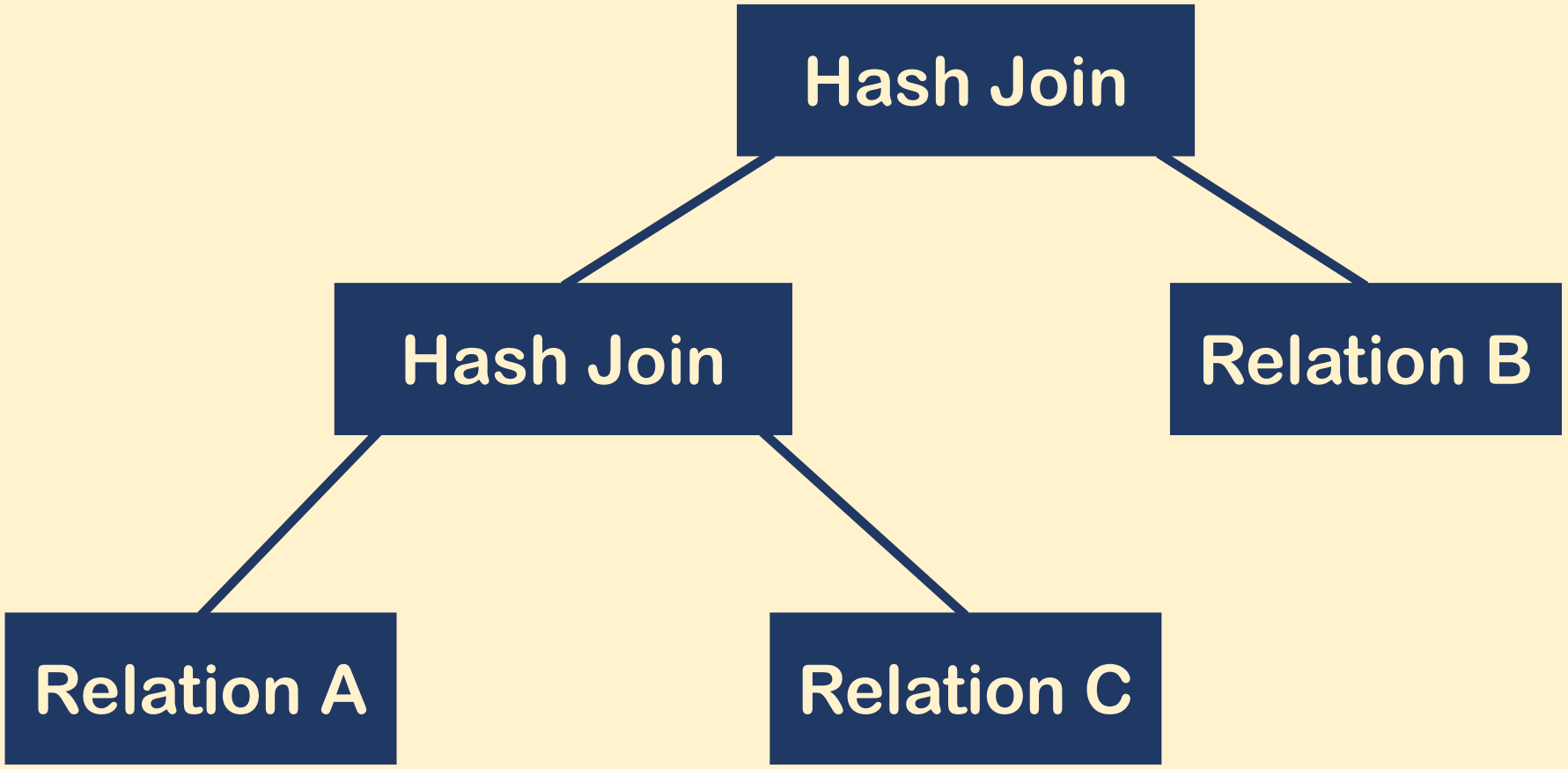
Avoid such explosive join



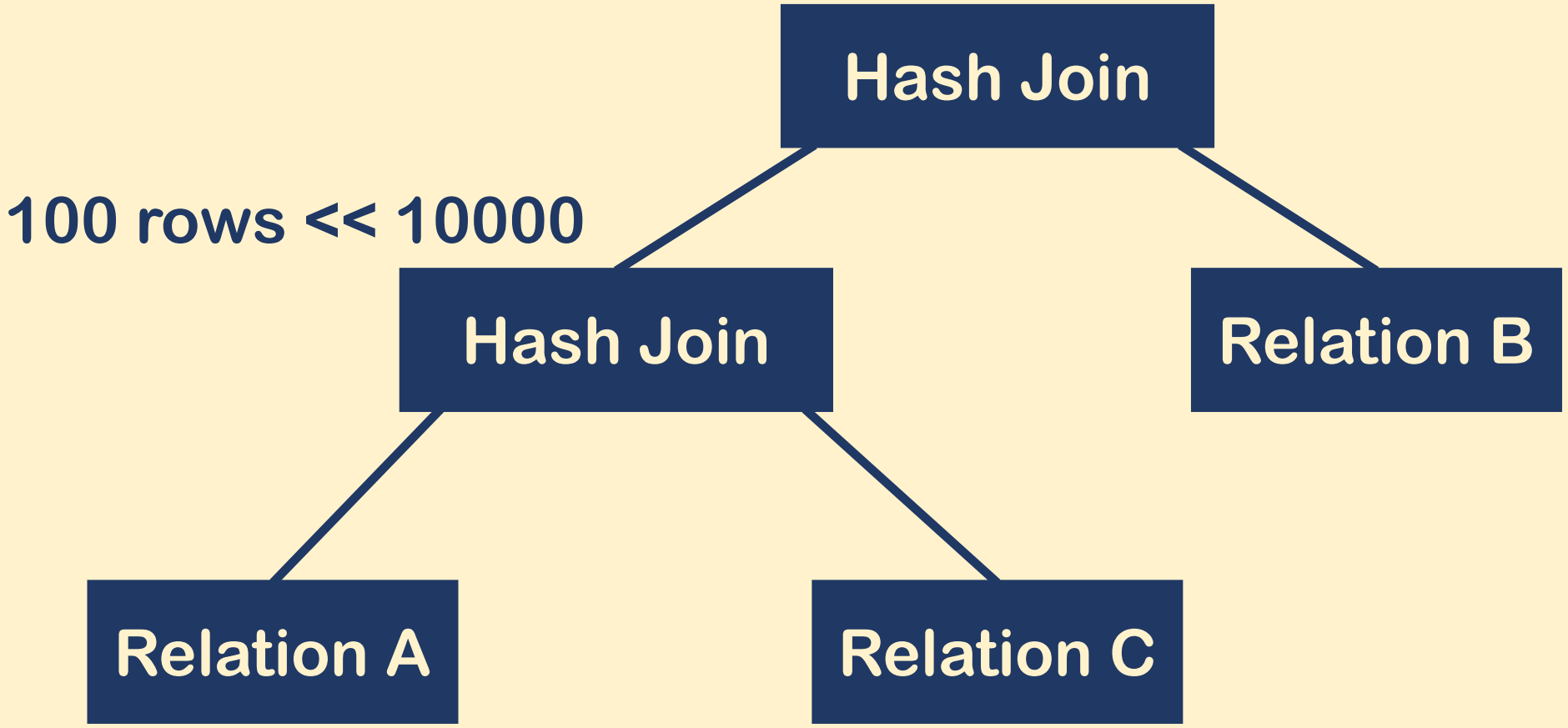
Avoid such explosive join



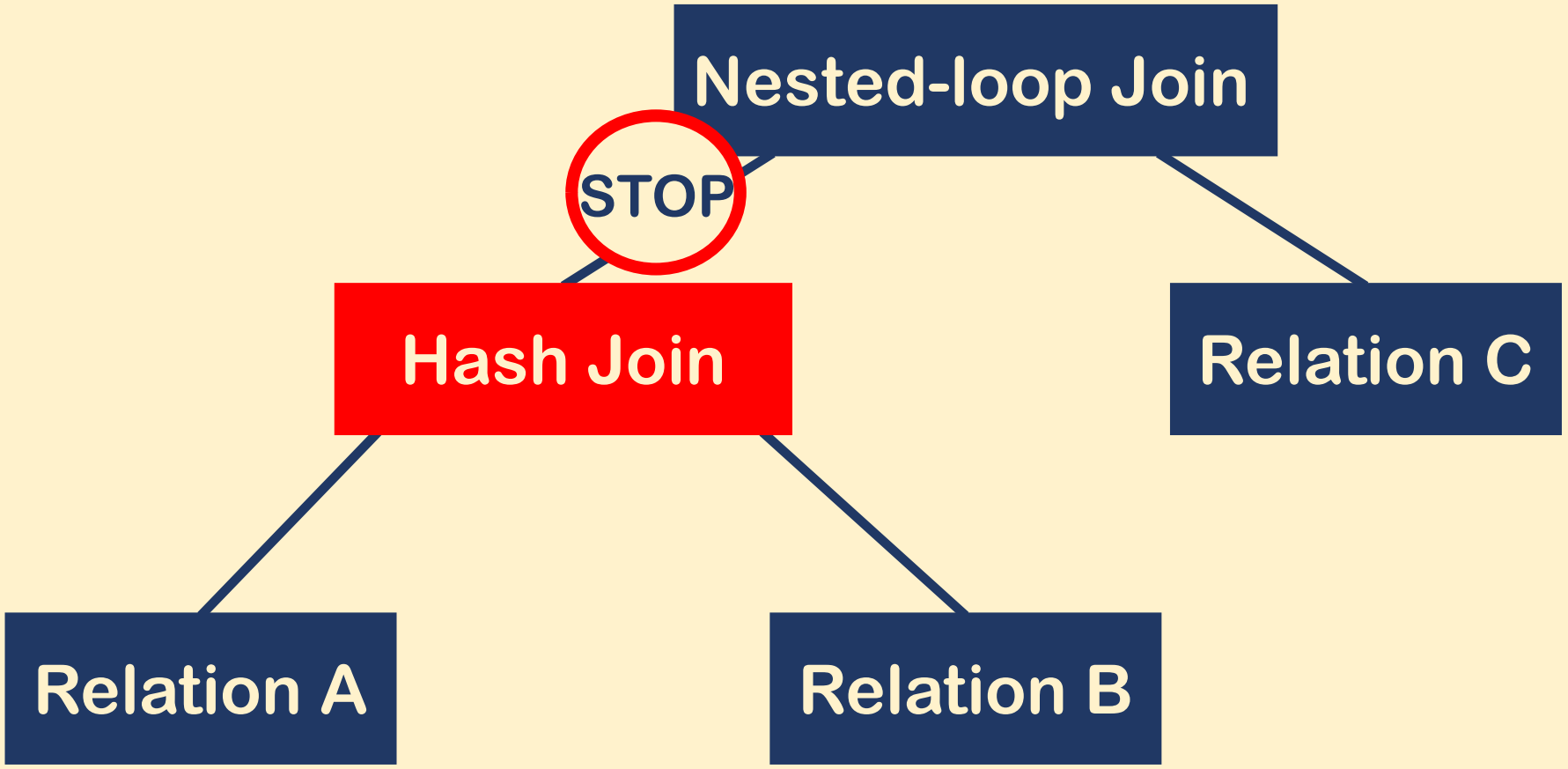
Avoid such explosive join



Avoid such explosive join

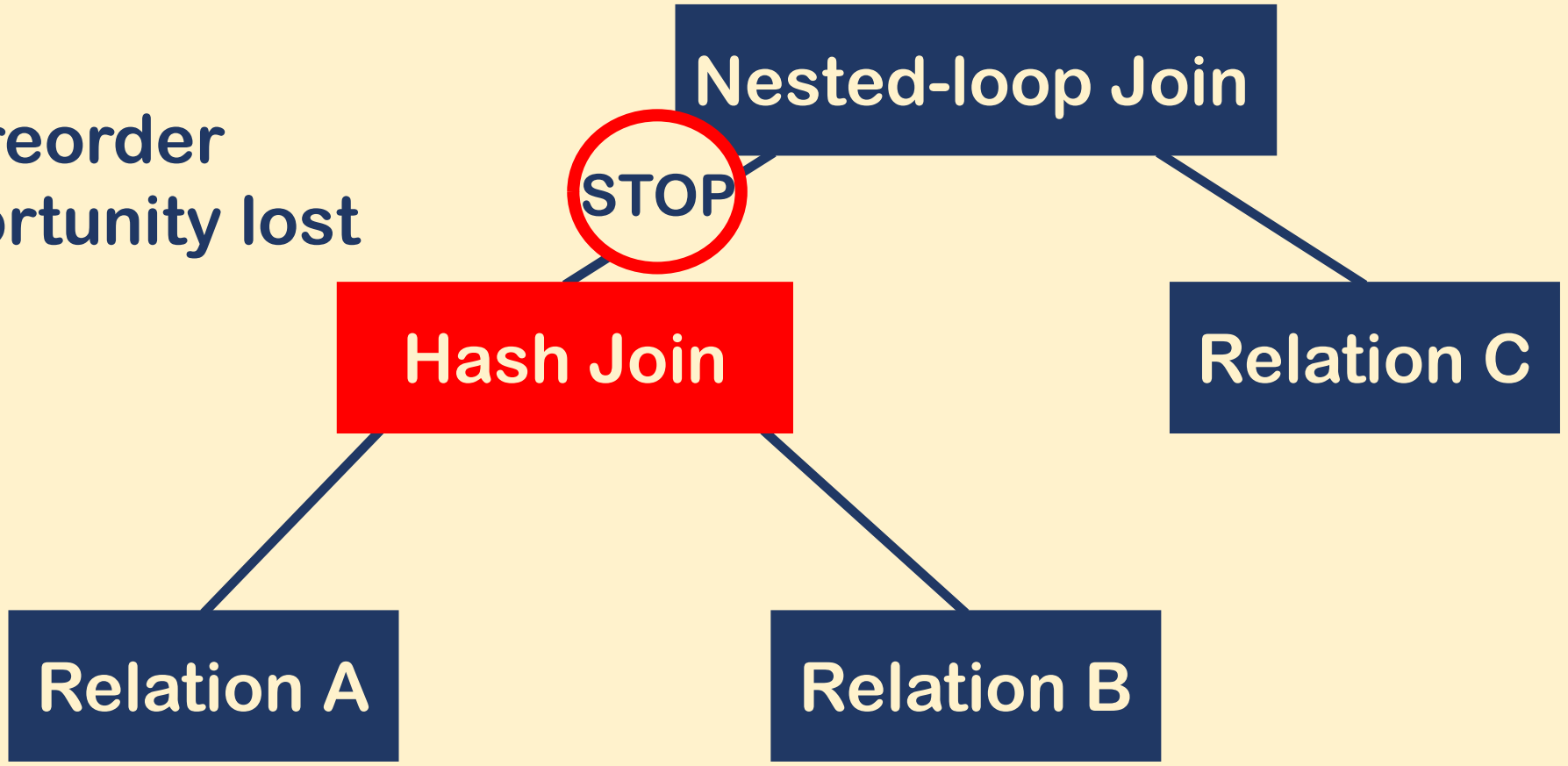


Current re-optimization cannot avoid this

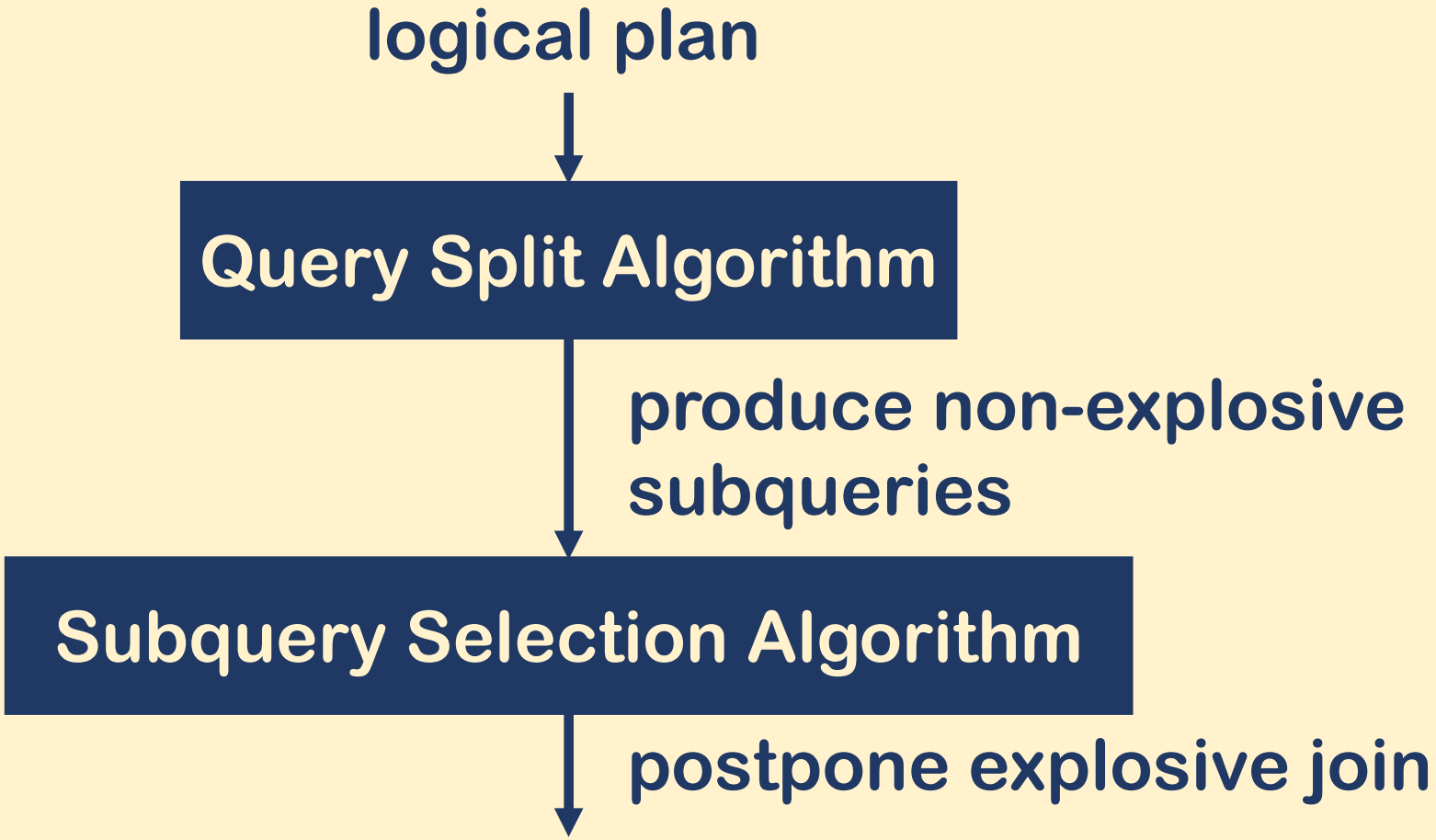


Current re-optimization cannot avoid this

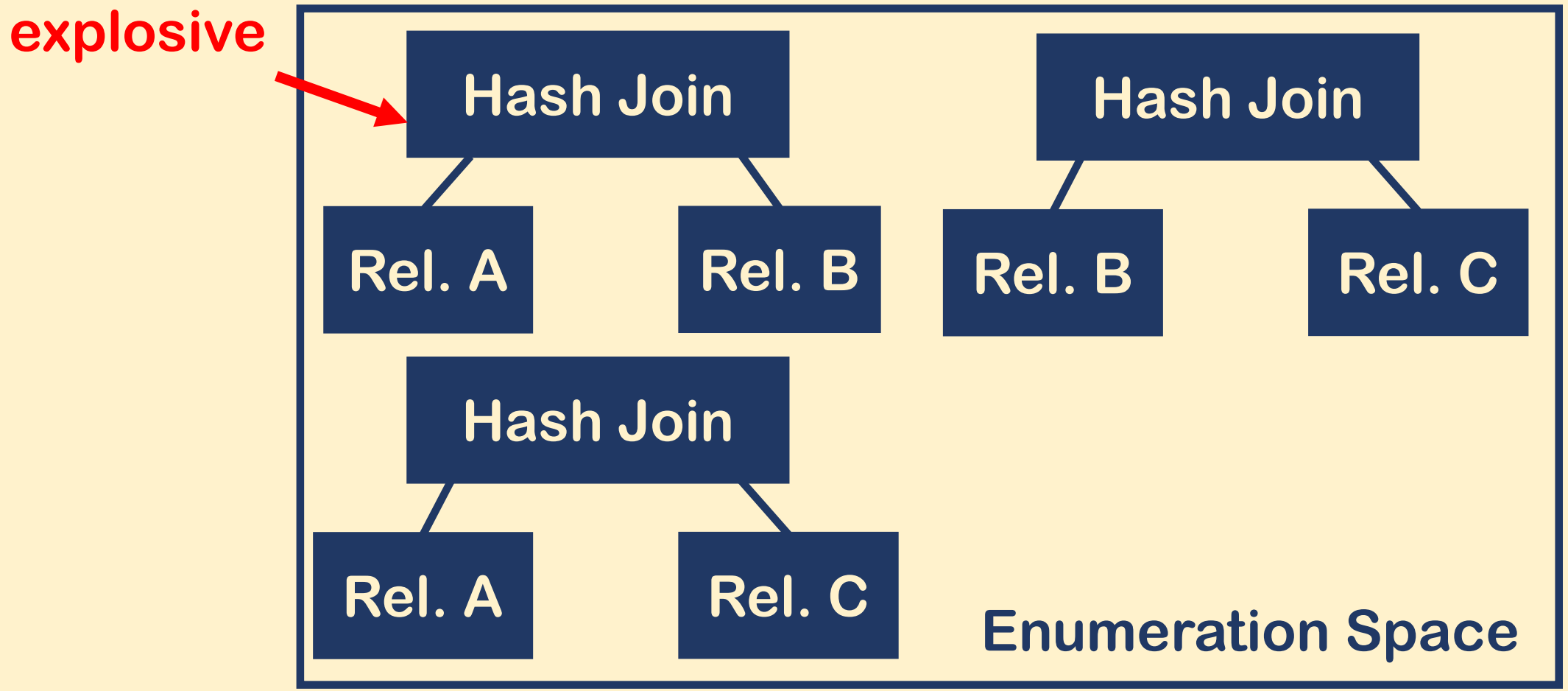
join reorder
opportunity lost



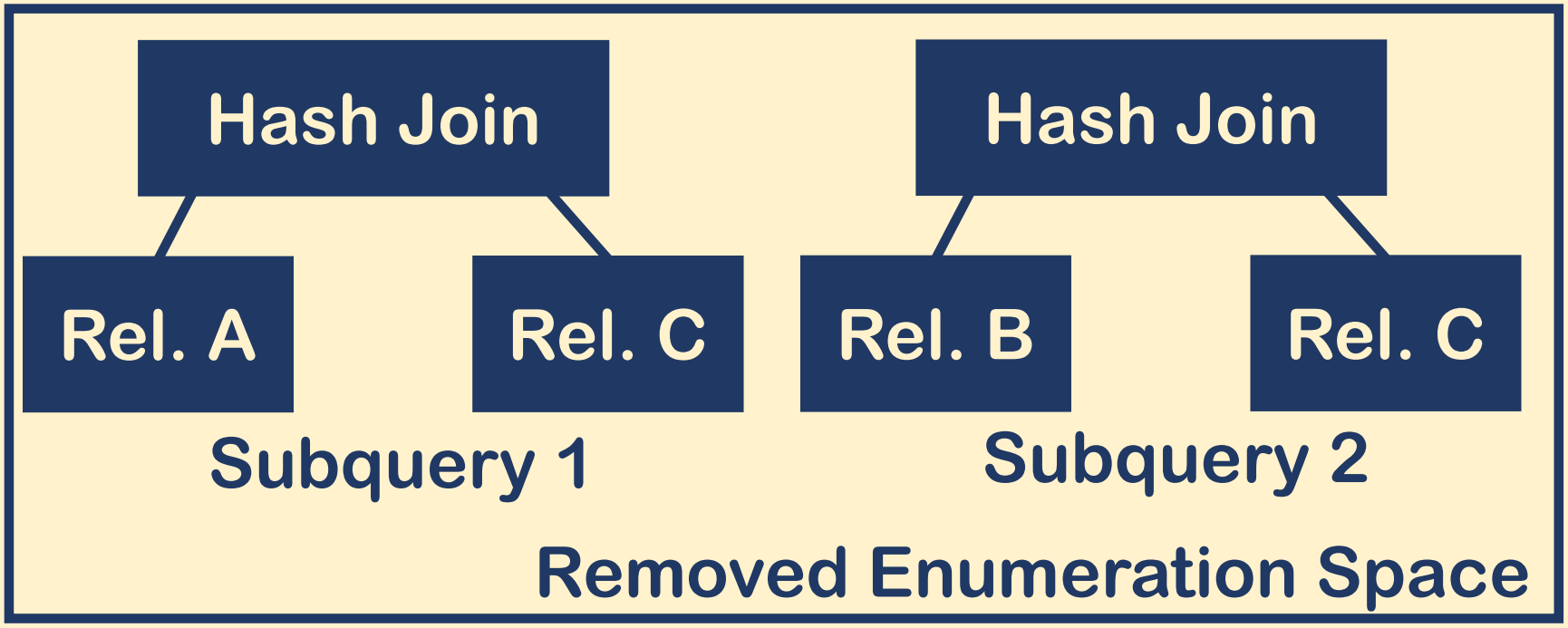
QuerySplit improve from two aspects



First, try to avoid explosive join

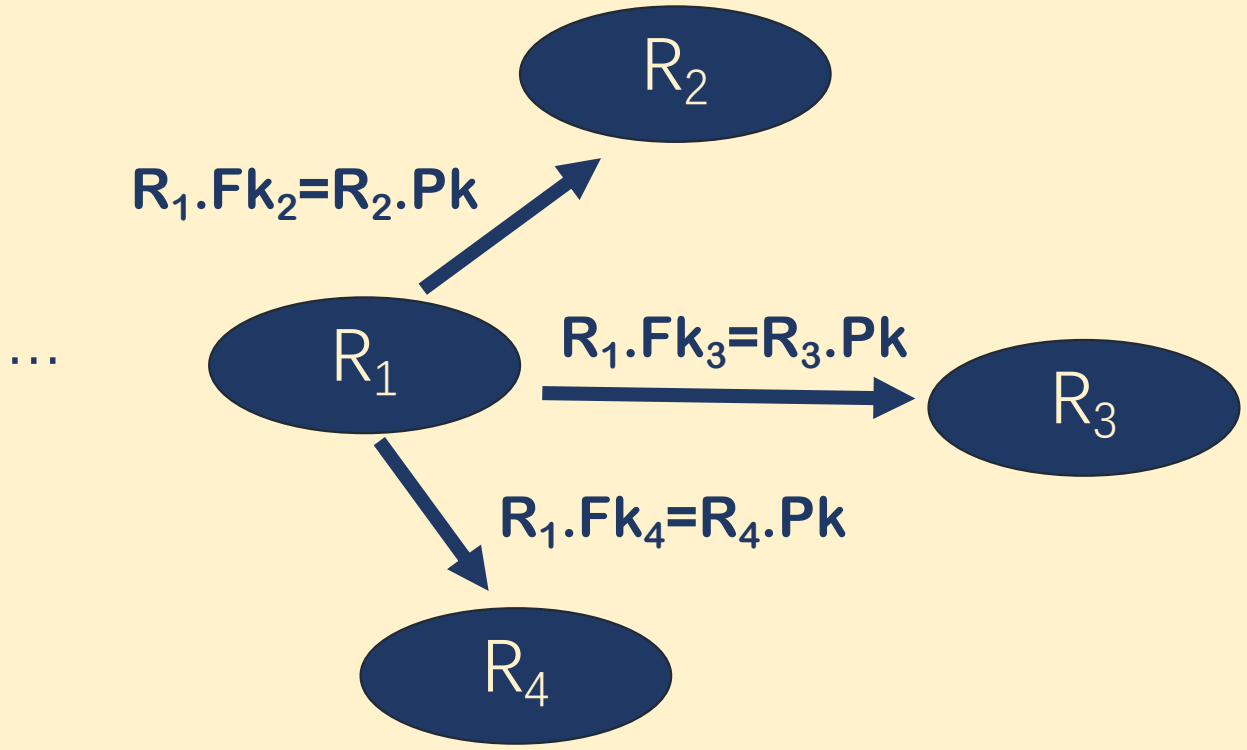


First, try to avoid explosive join

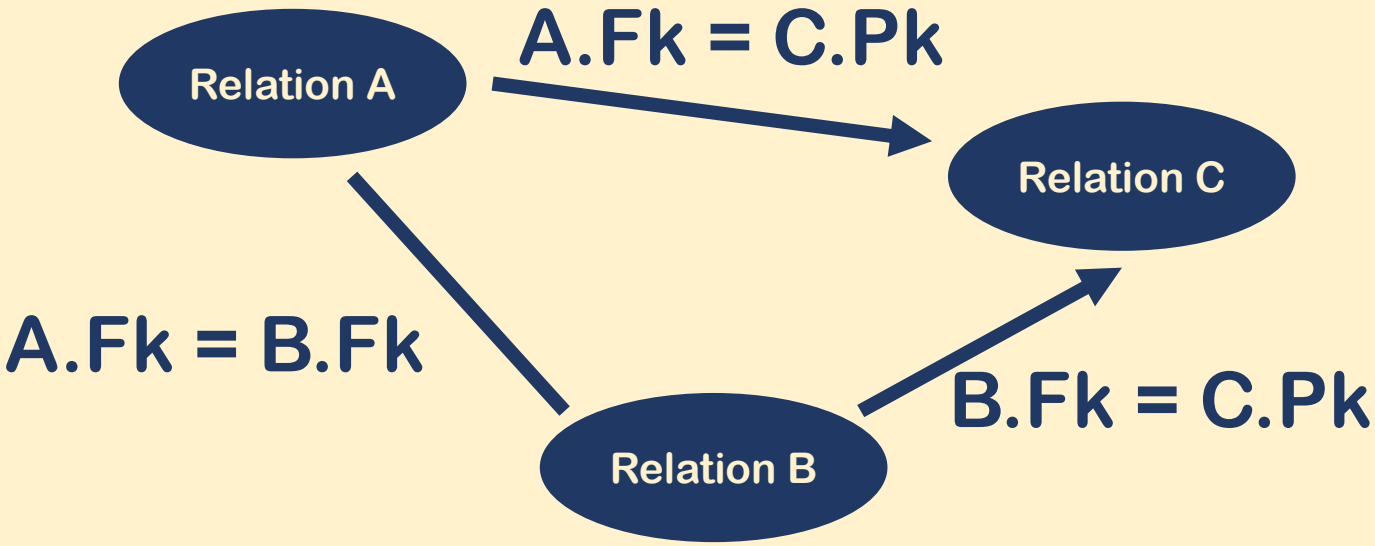


Query Split Algorithm avoids explosive join in advance

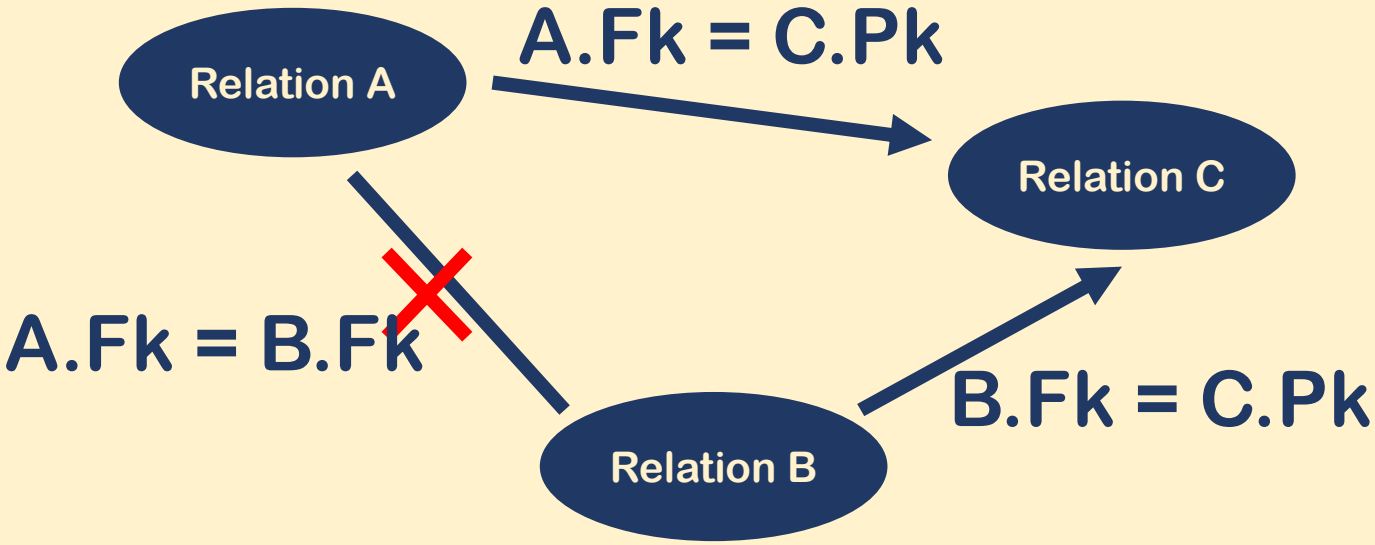
Fk-Pk join constrains the result size



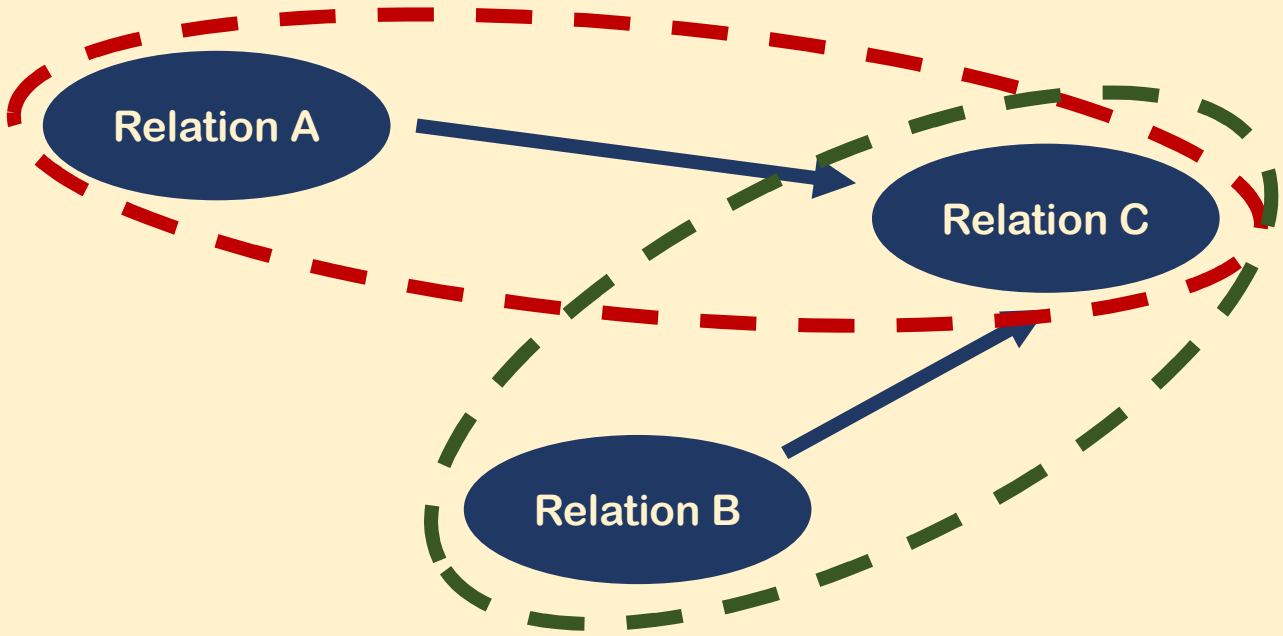
First, try to avoid explosive join



First, try to avoid explosive join

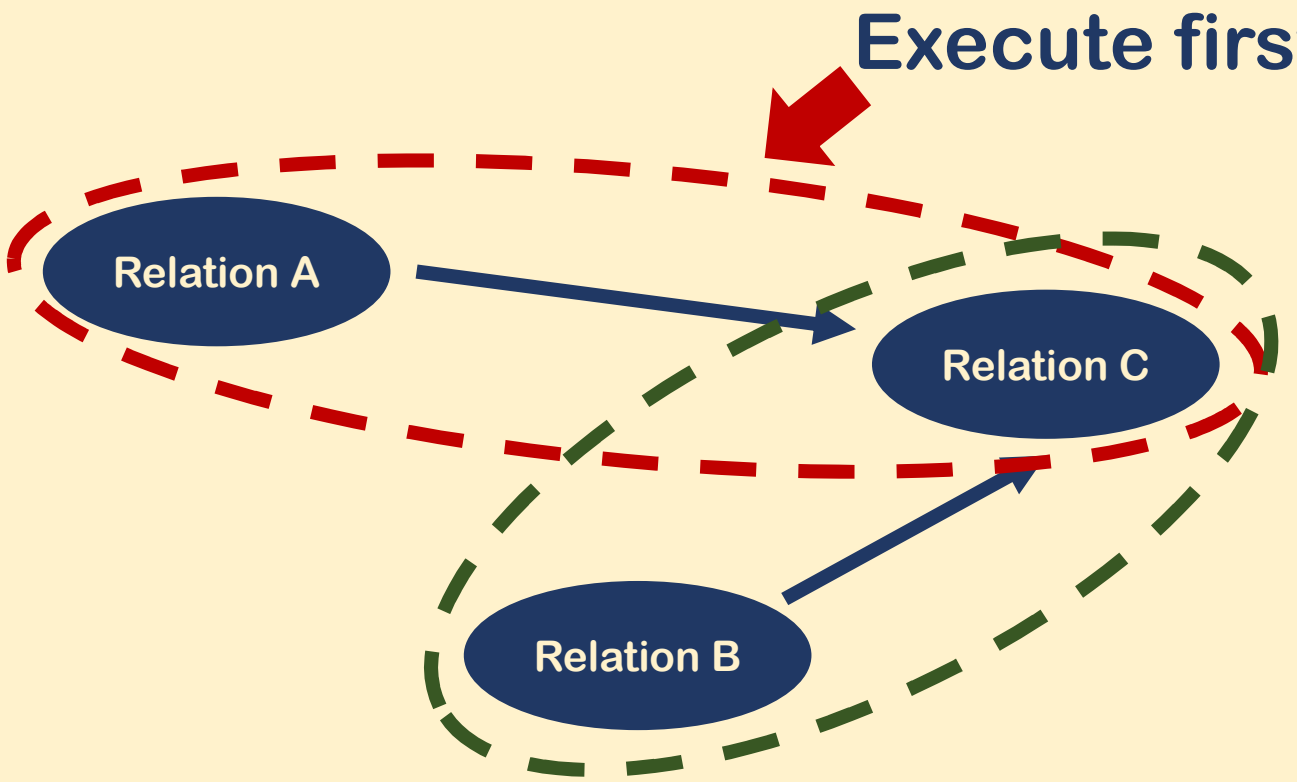


First, try to avoid explosive join



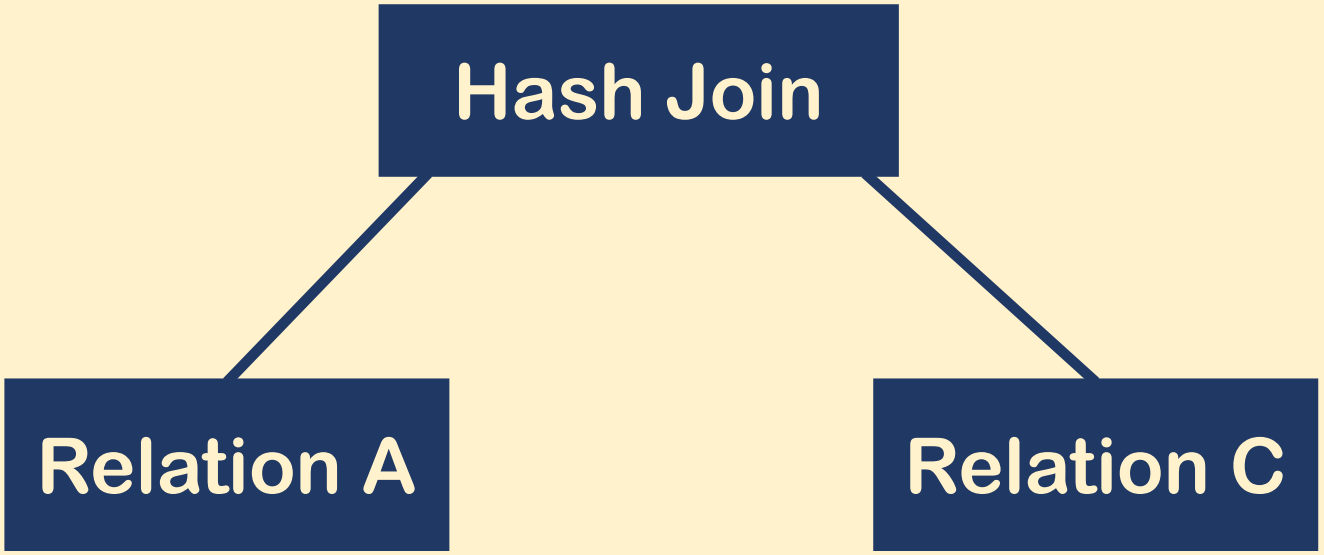
FK Center

First, try to avoid explosive join

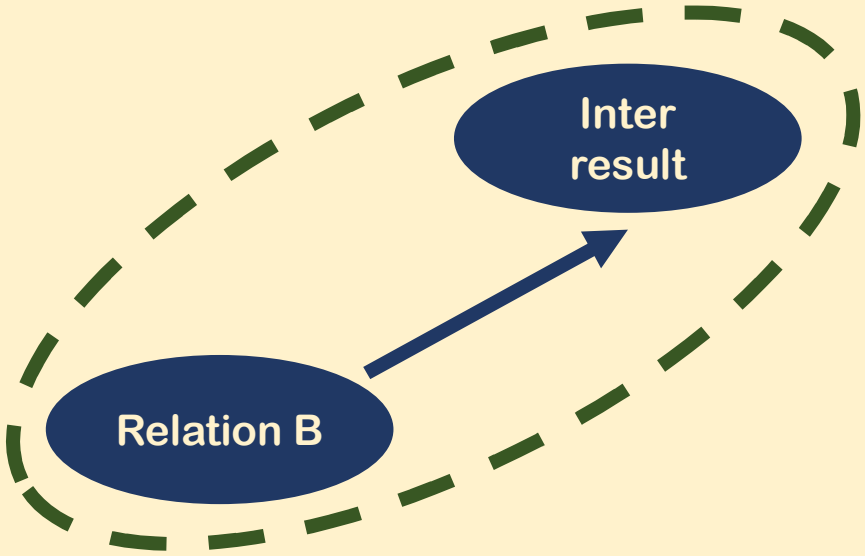


First, try to avoid explosive join

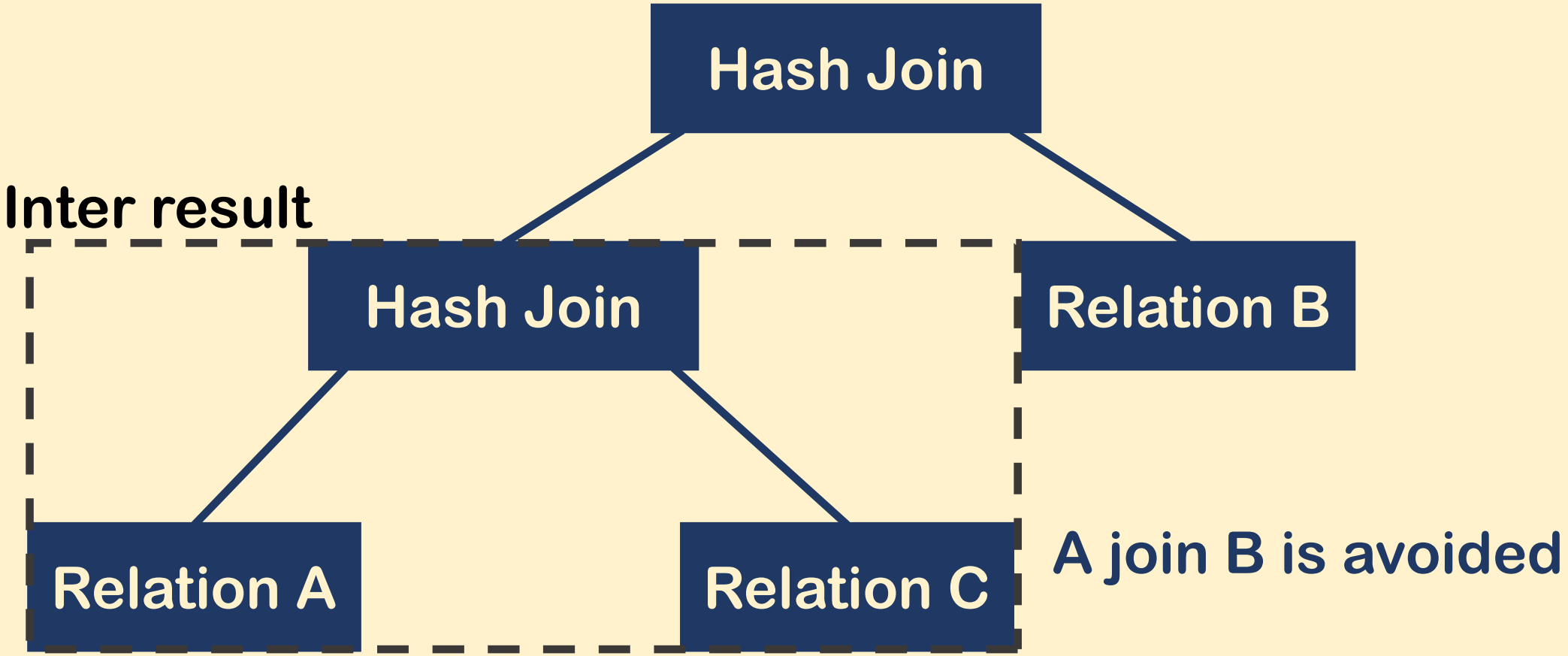
100 rows << 10000



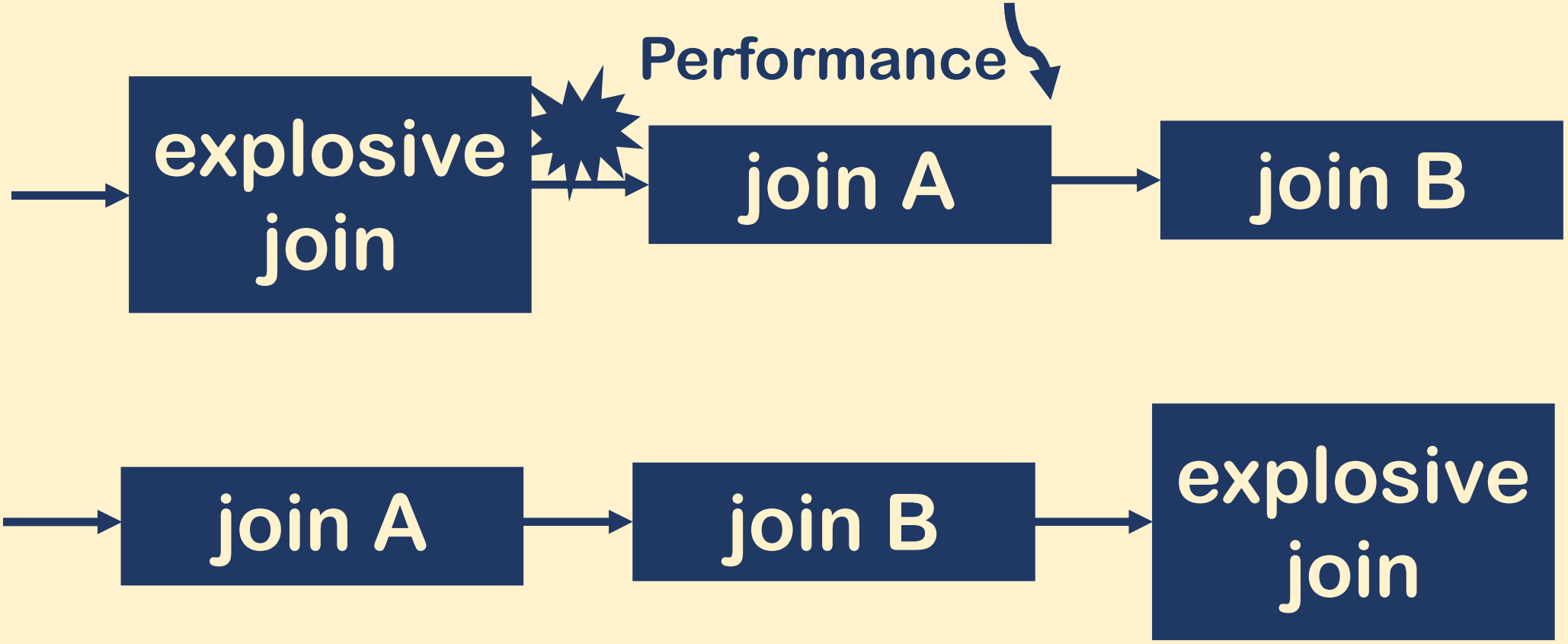
First, try to avoid explosive join



First, try to avoid explosive join

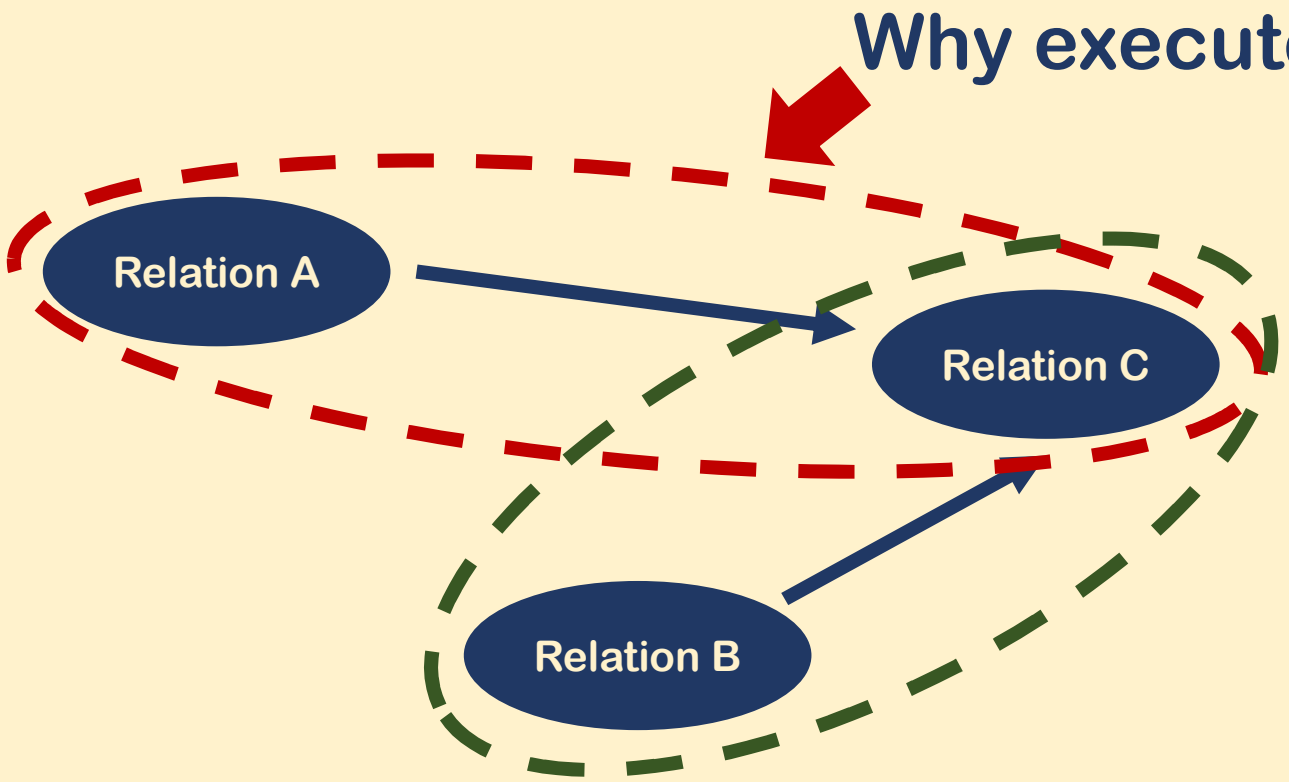


Second, postpone explosive join



Subquery Selection Algorithm delays explosive join

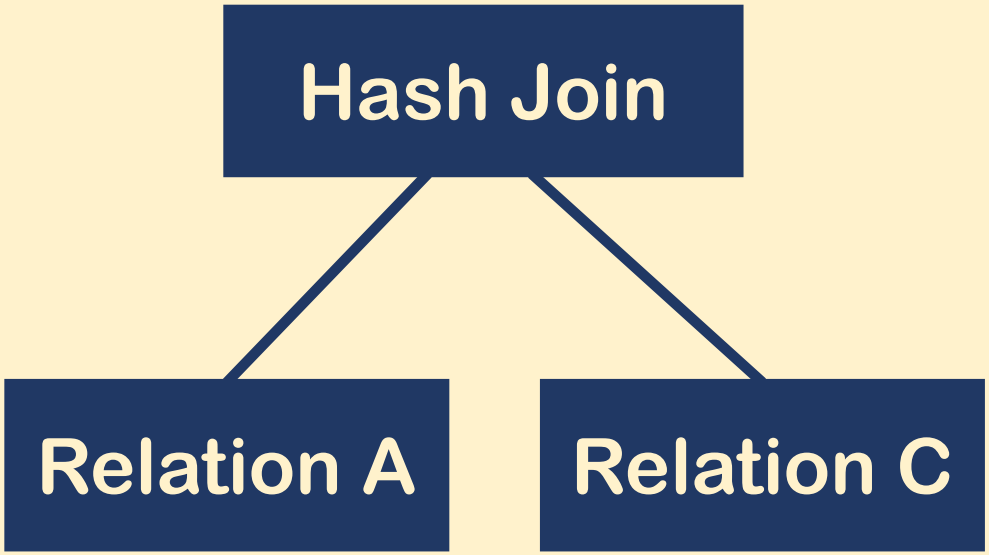
Associated with execution order



Prefer executing small subquery

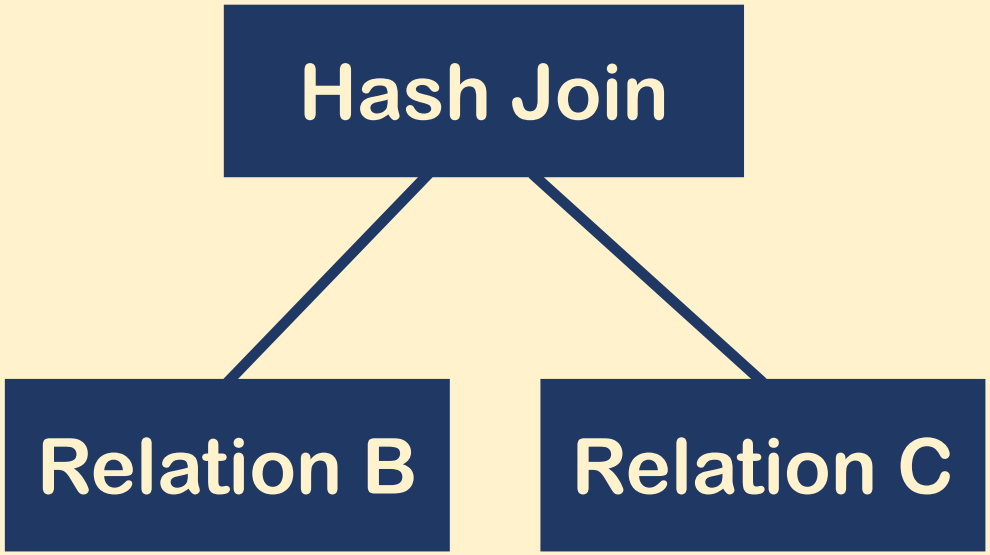
output rows: x_1

execution time: y_1



output rows: x_2

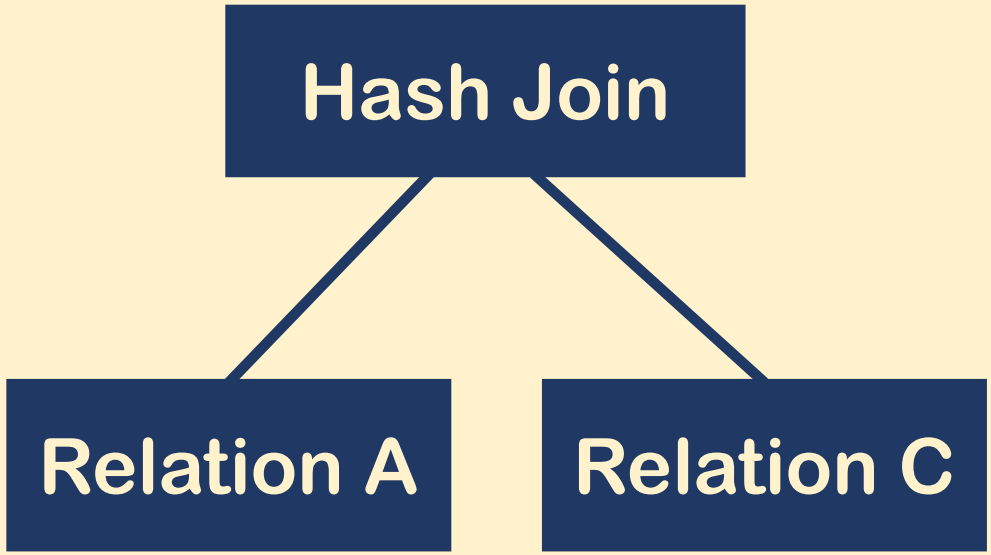
execution time: y_2



Prefer executing small subquery

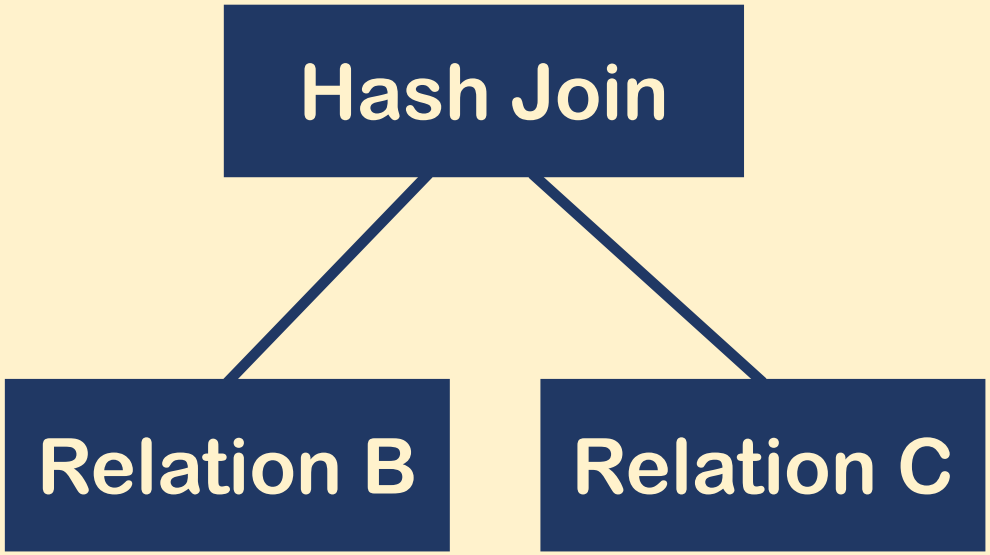
output rows: x_1

execution time: y_1



output rows: x_2

execution time: y_2



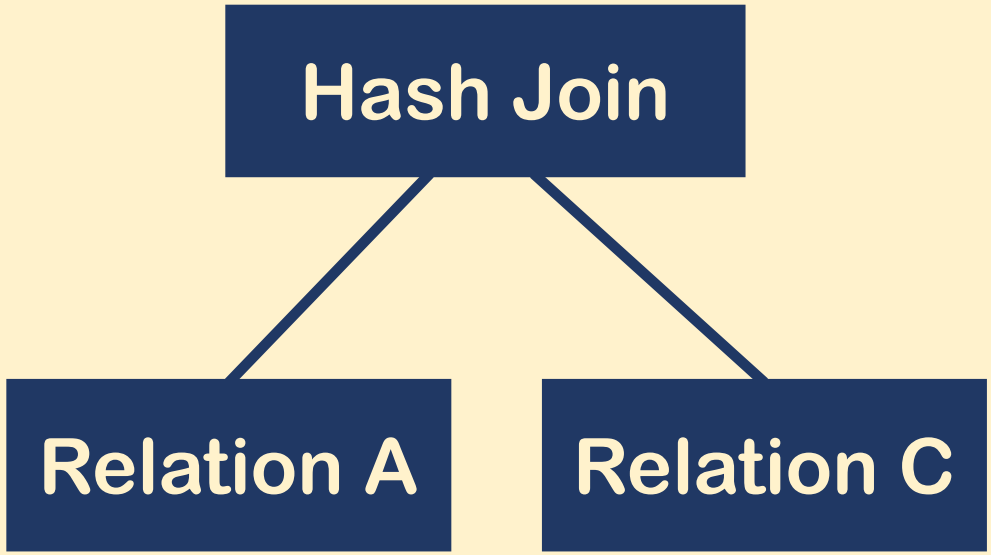
$x_1 < x_2$

$y_1 < y_2$

Prefer executing small subquery

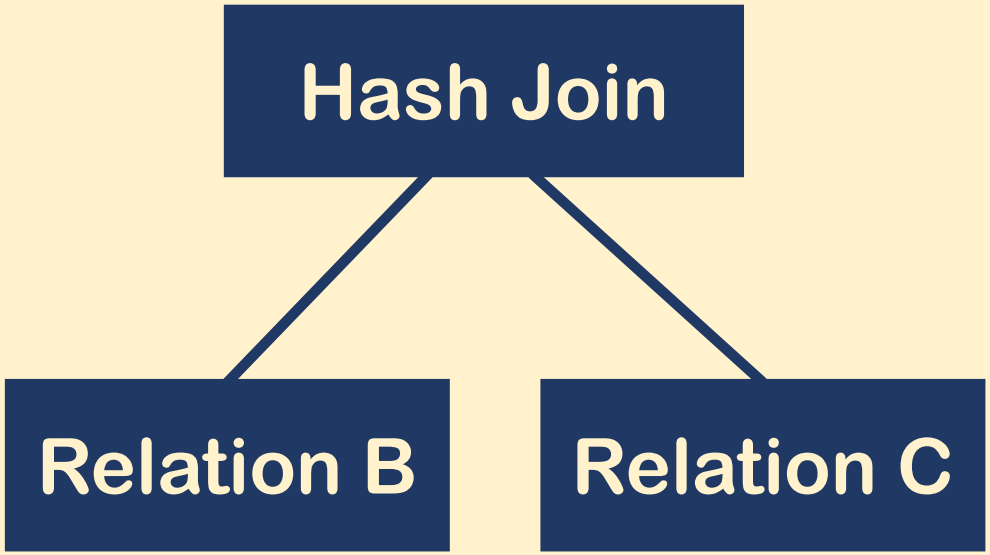
output rows: x_1

execution time: y_1



output rows: x_2

execution time: y_2



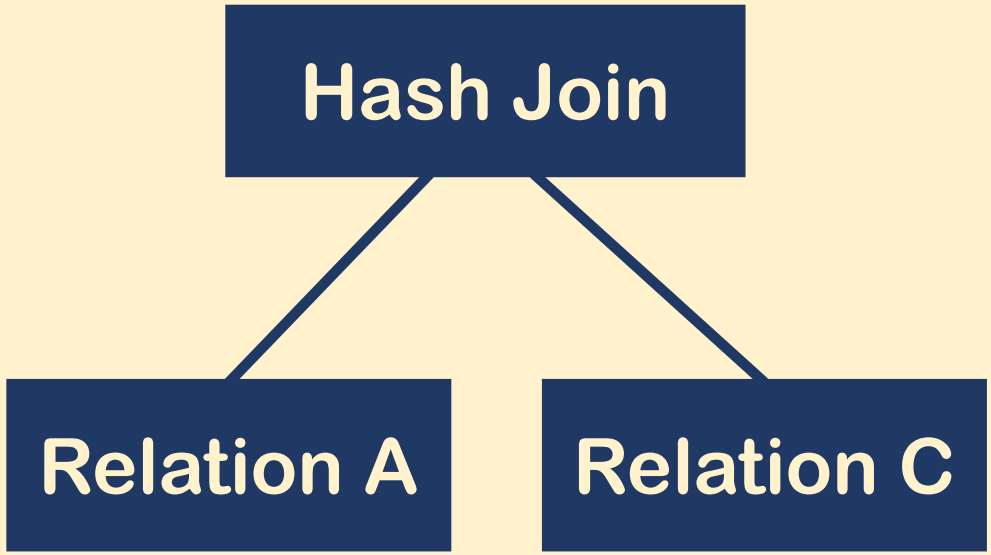
$x_1 < x_2$

$y_1 > y_2$?

Prefer executing small subquery

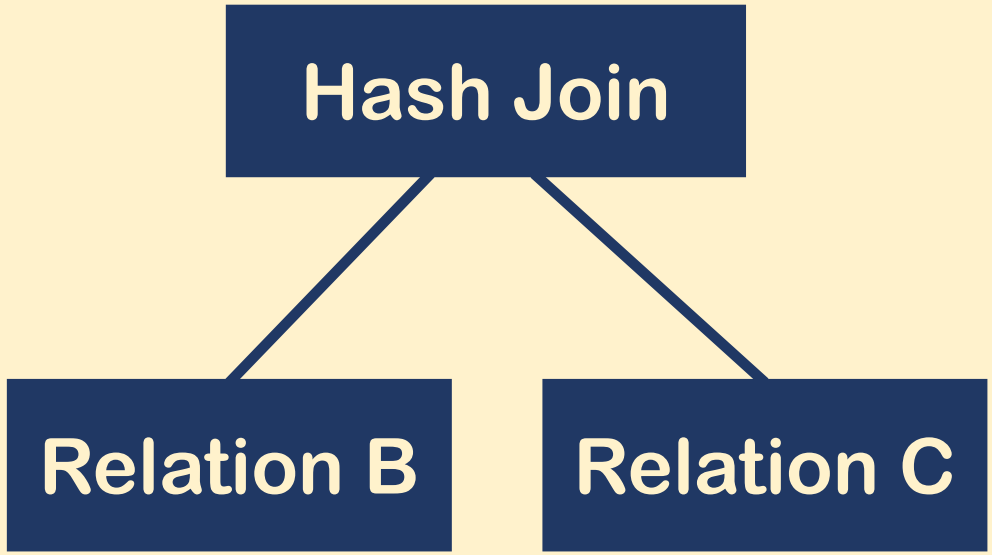
output rows: x_1

execution time: y_1



output rows: x_2

execution time: y_2



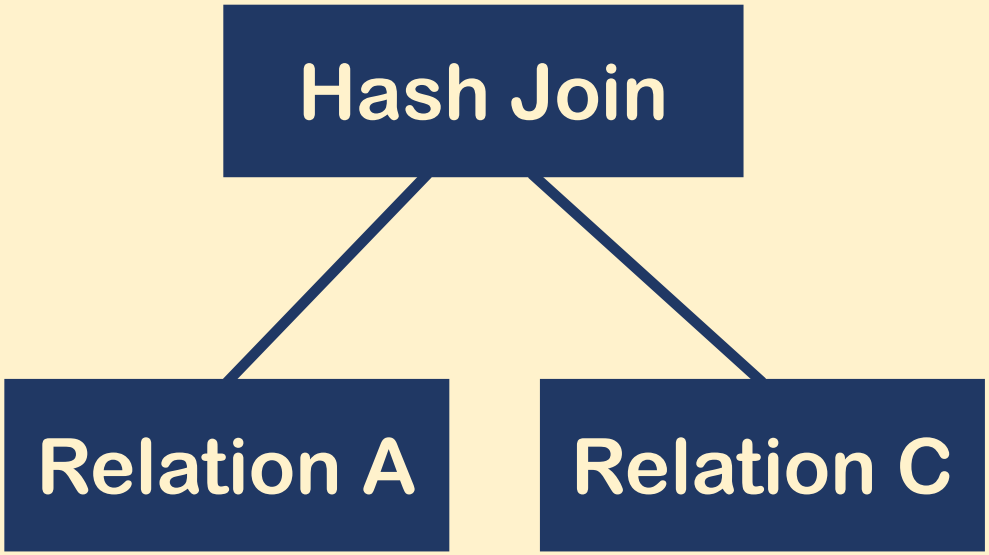
$x_1 < x_2$ Future?

$y_1 > y_2$ Current?

Prefer executing small subquery

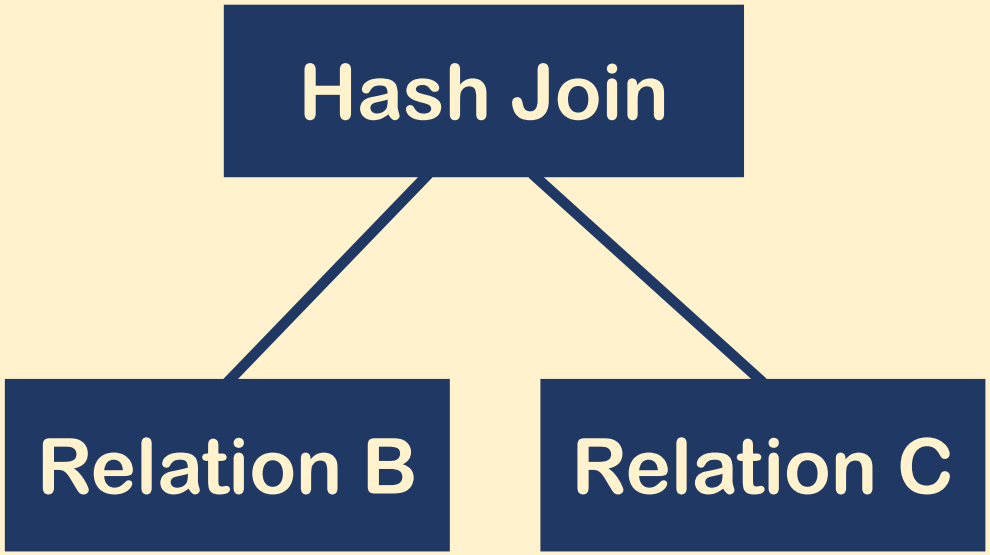
output rows: x_1

execution time: y_1



output rows: x_2

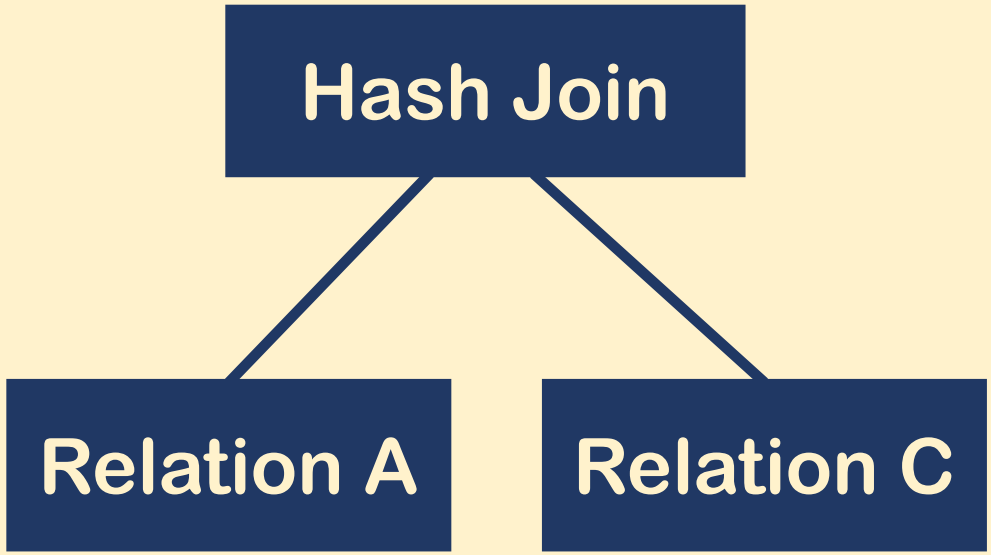
execution time: y_2



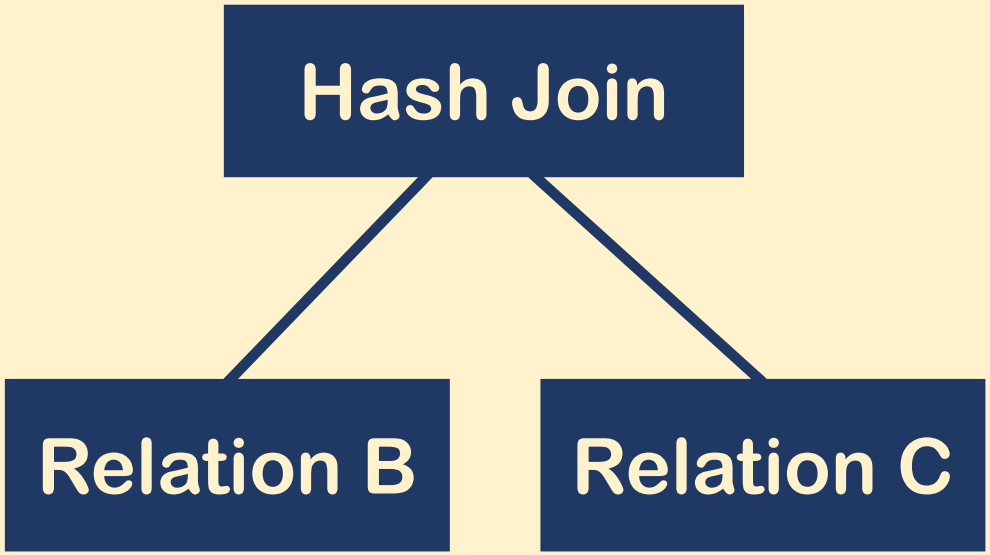
$f(x_1, y_1)$? $f(x_2, y_2)$

Prefer executing small subquery

output rows: x_1
execution time: y_1



output rows: x_2
execution time: y_2



$f(x_1, y_1) < f(x_2, y_2)$

Prefer executing small subquery

Φ_1	x
Φ_2	log(x) * y
Φ_3	sqrt(x) * y
Φ_4	x * y
Φ_5	y

output rows: x

execution time: y

Evaluation Setup: a real-world workload

Workload

- **JOB (main)**
- **DSB**
- **TPC-H**

Evaluation Setup: a real-world workload

Workload

- JOB (main)
- TPC-H
- DSB

System Config

- Windows 10
- 128 GB Memory

Evaluation Setup: a real-world workload

Workload

- JOB (main)
- DSB
- TPC-H

System Config

- Windows 10
- 128 GB Memory

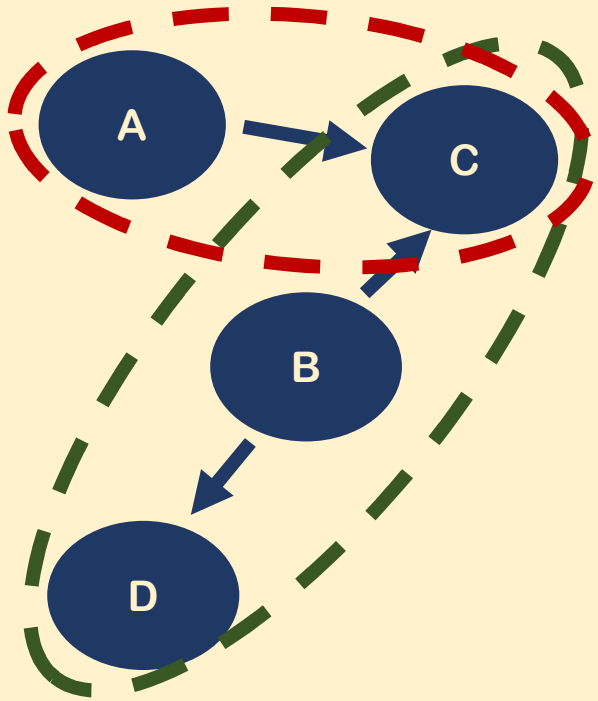
Database Config

- PostgreSQL
- No parallelization
- 8GB effective cache
- 1000s Timeout

Best Implementation for QuerySplit

FK-Center vs. other 2 strategies

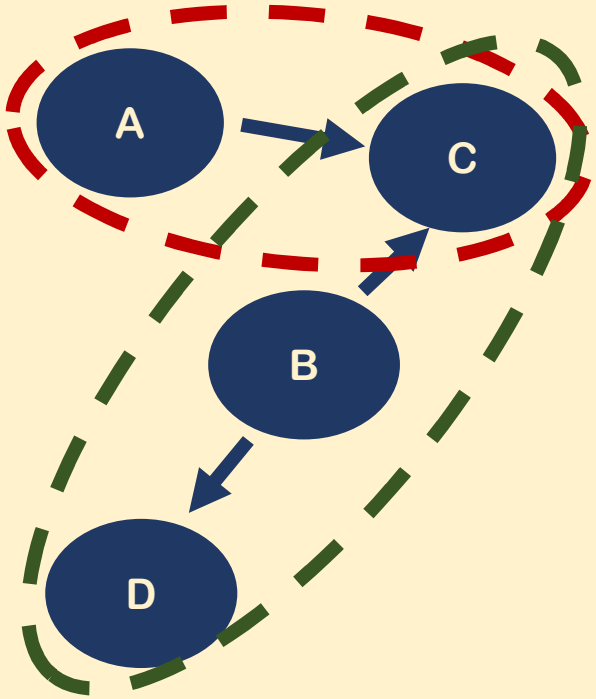
FK-Center



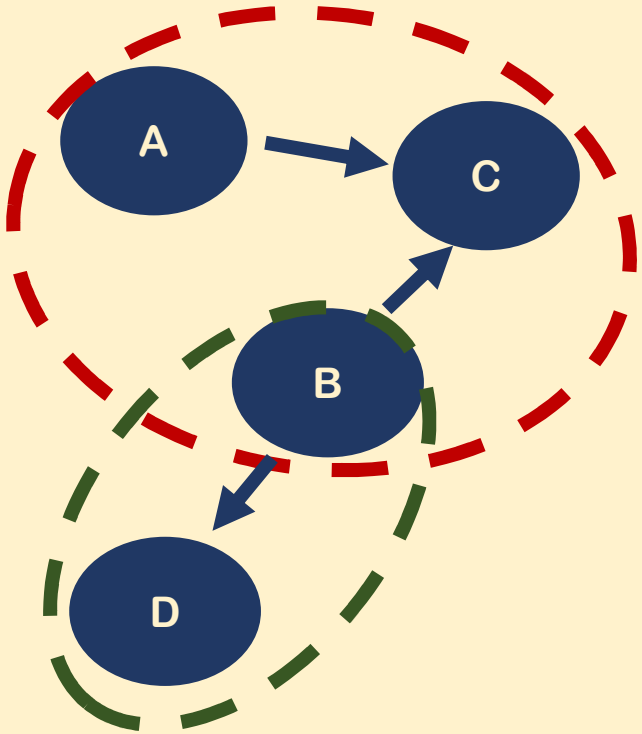
Best Implementation for QuerySplit

FK-Center vs. other 2 strategies

FK-Center



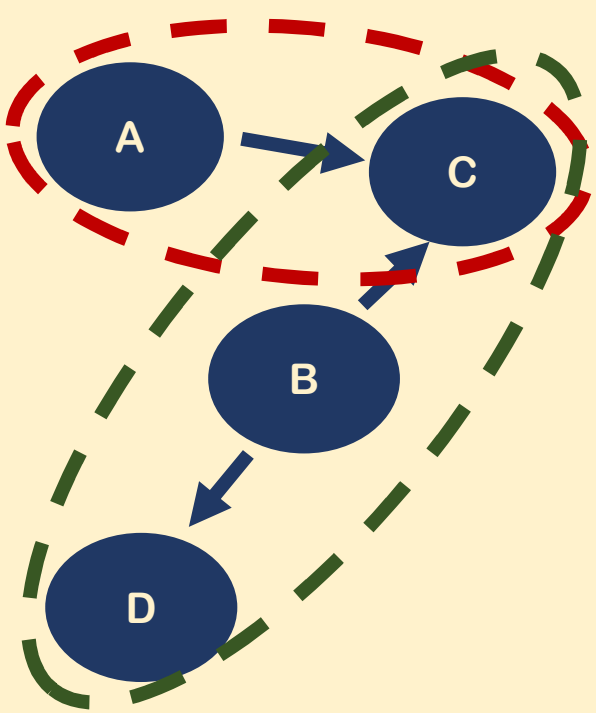
PK-Center



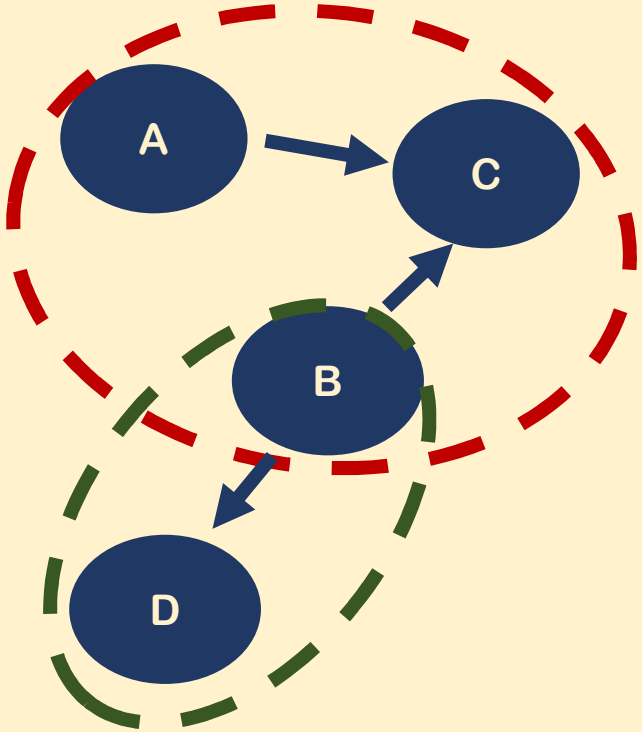
Best Implementation for QuerySplit

FK-Center vs. other 2 strategies

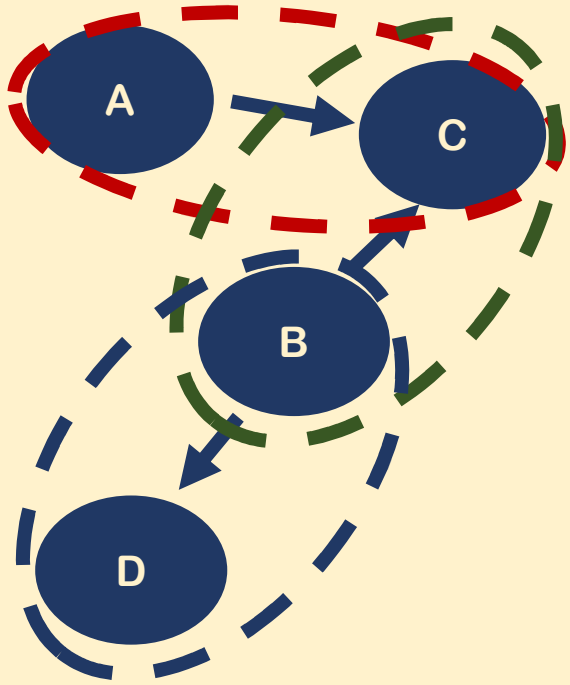
FK-Center



PK-Center



MinSubquery



Best Implementation for QuerySplit

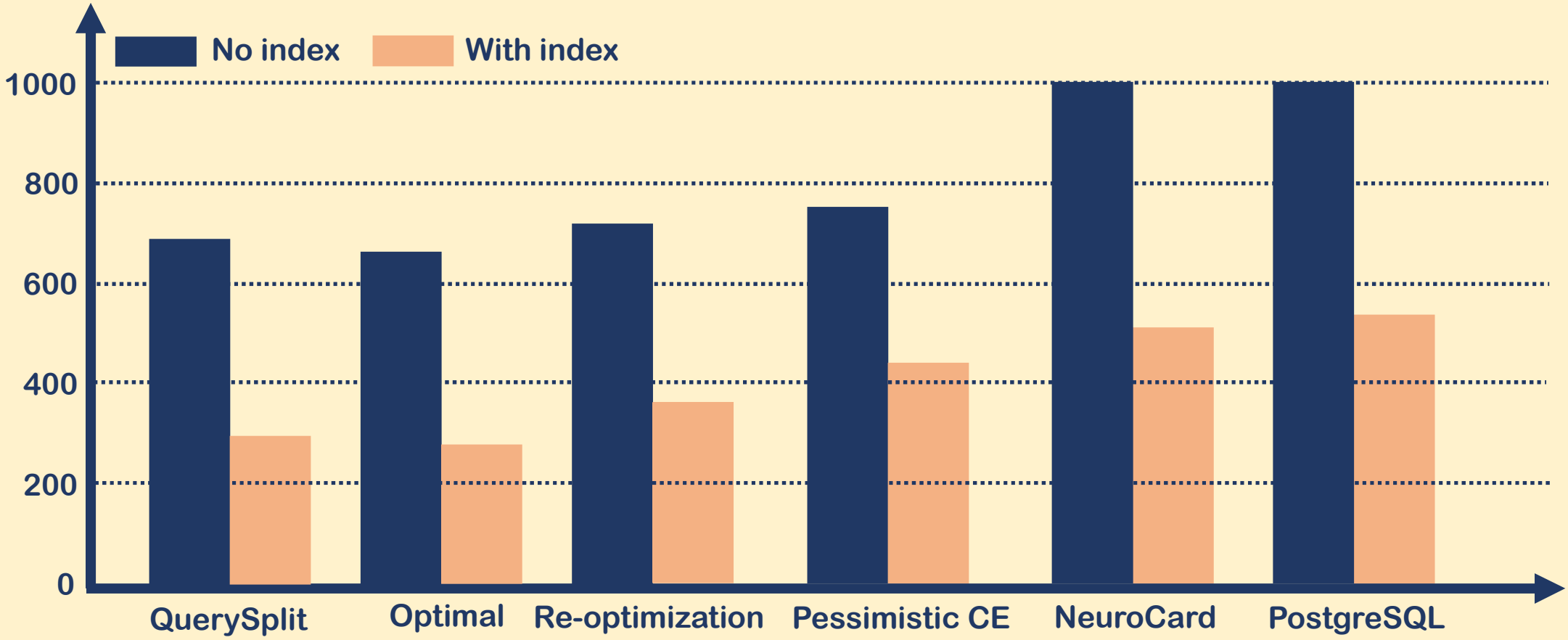
	FK-Center	PK-Center	MinSubquery
Φ_1 = x	421s	378s	463s
Φ_2 = log(x) * y	327s	349s	428s
Φ_3 = sqrt(x) * y	328s	339s	418s
Φ_4 = x * y	295s	350s	427s
Φ_5 = y	348s	407s	474s

Best Implementation for QuerySplit

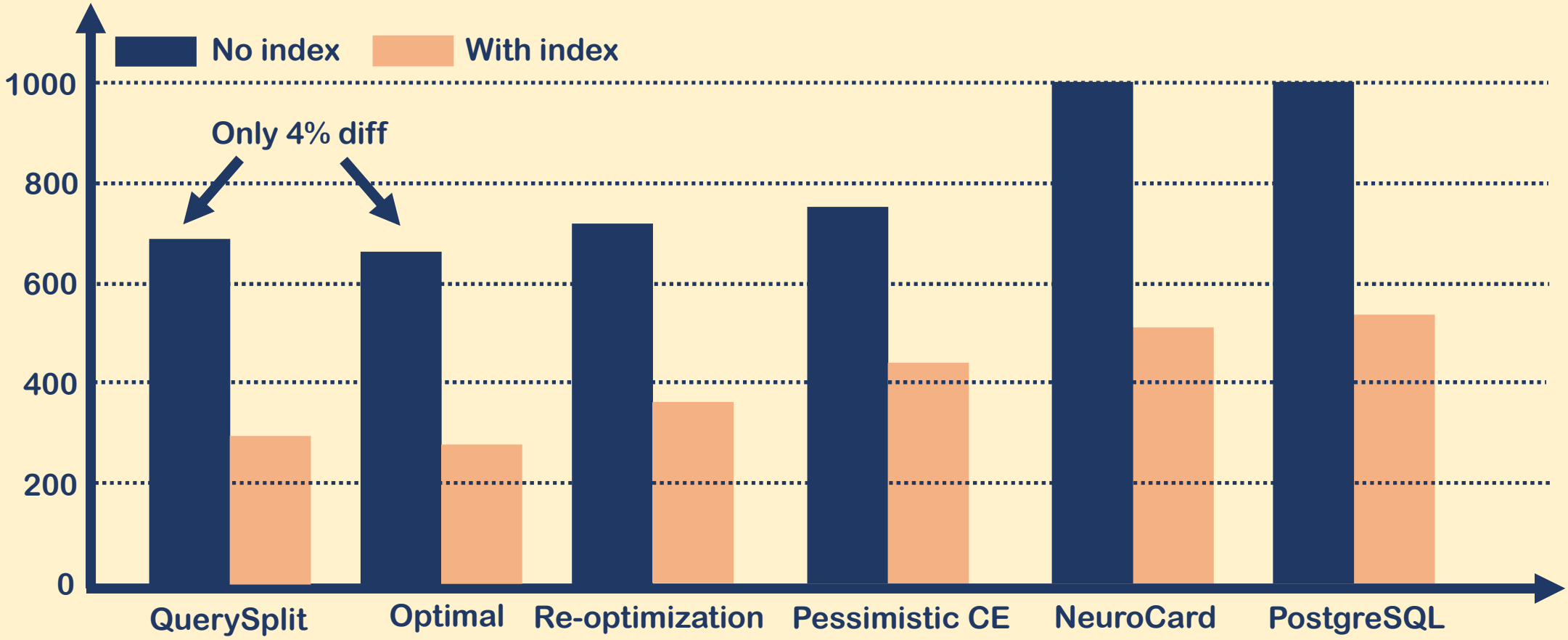
	FK-Center	PK-Center	MinSubquery
Φ_1 = x	421s	378s	463s
Φ_2 = log(x) * y	327s	349s	428s
Φ_3 = sqrt(x) * y	328s	339s	418s
Φ_4 = x * y	2	350s	427s
Φ_5 = y	348s	407s	474s

Best

QuerySplit speeds up end-to-end latency



QuerySplit speeds up end-to-end latency



Conclusion

- Current re-optimization can be misled by the initial physical plan
- Two key ideas of QuerySplit
  - Query Split Algorithm produces non-explosive subuquery
  - Subquery Selection Algorithm postpones the inevitable explosive join