HARVARD
School of Engineering
and Applied Sciences

CS165
Data Systems

Have fun learning to design and build modern data systems

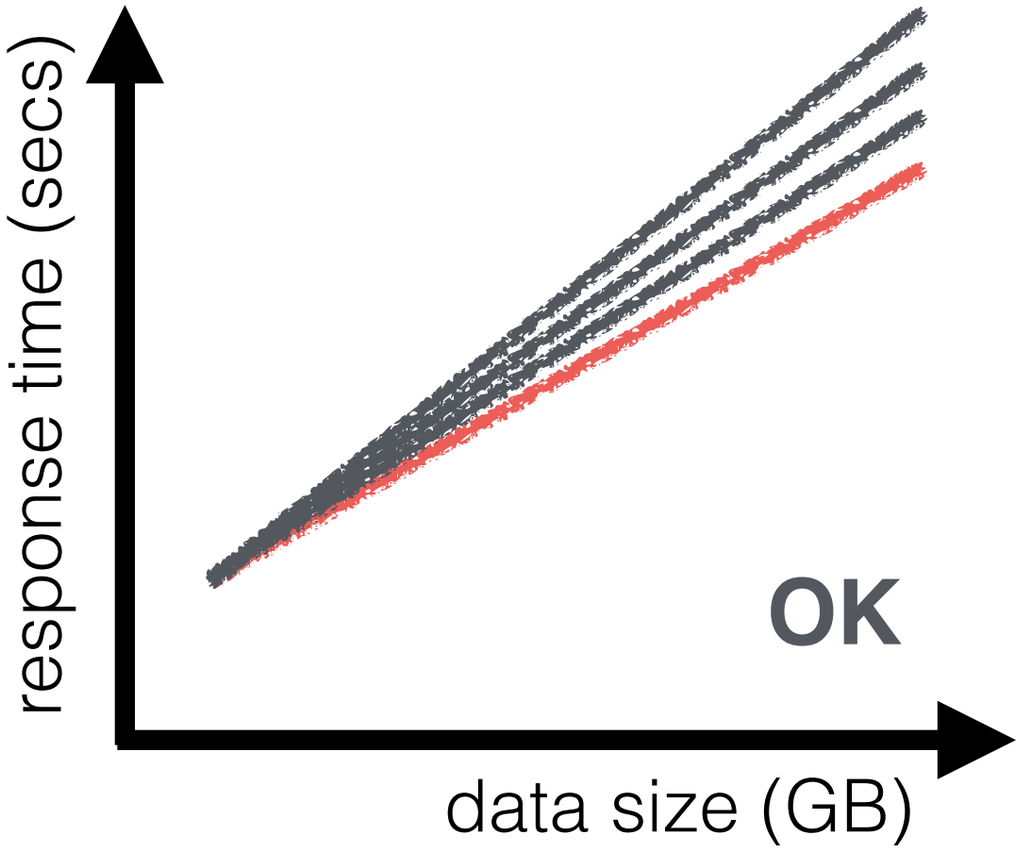
class 15

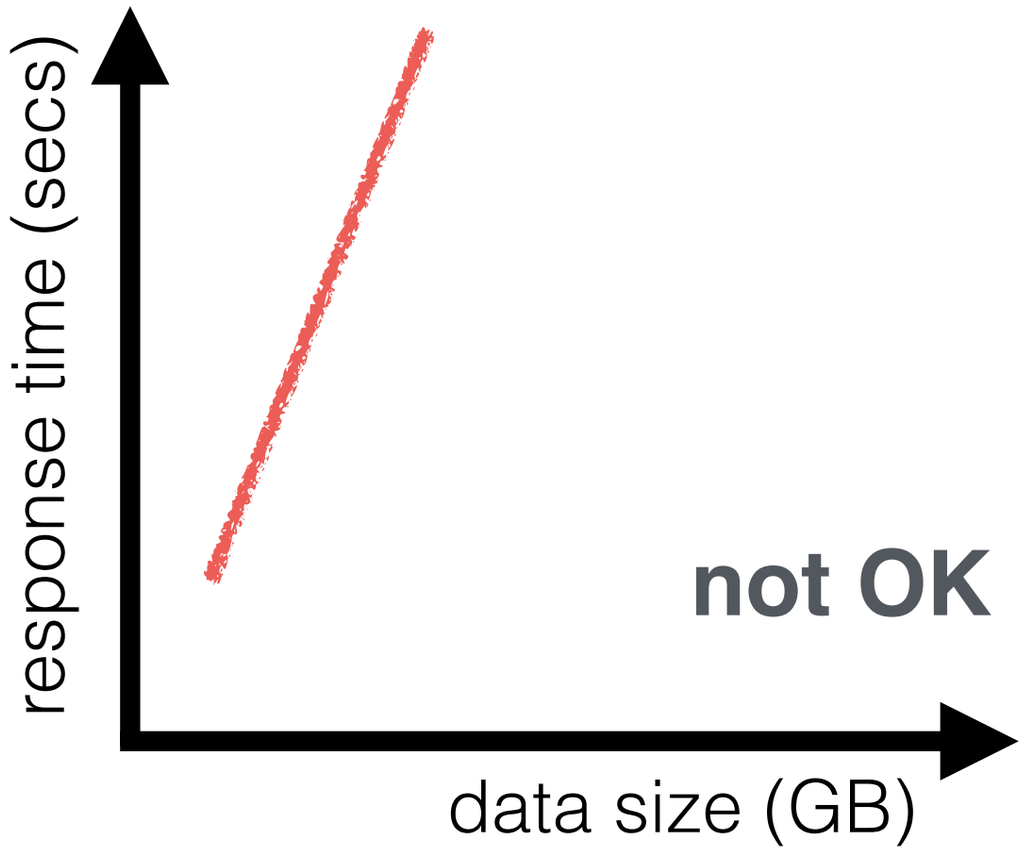
joins

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/

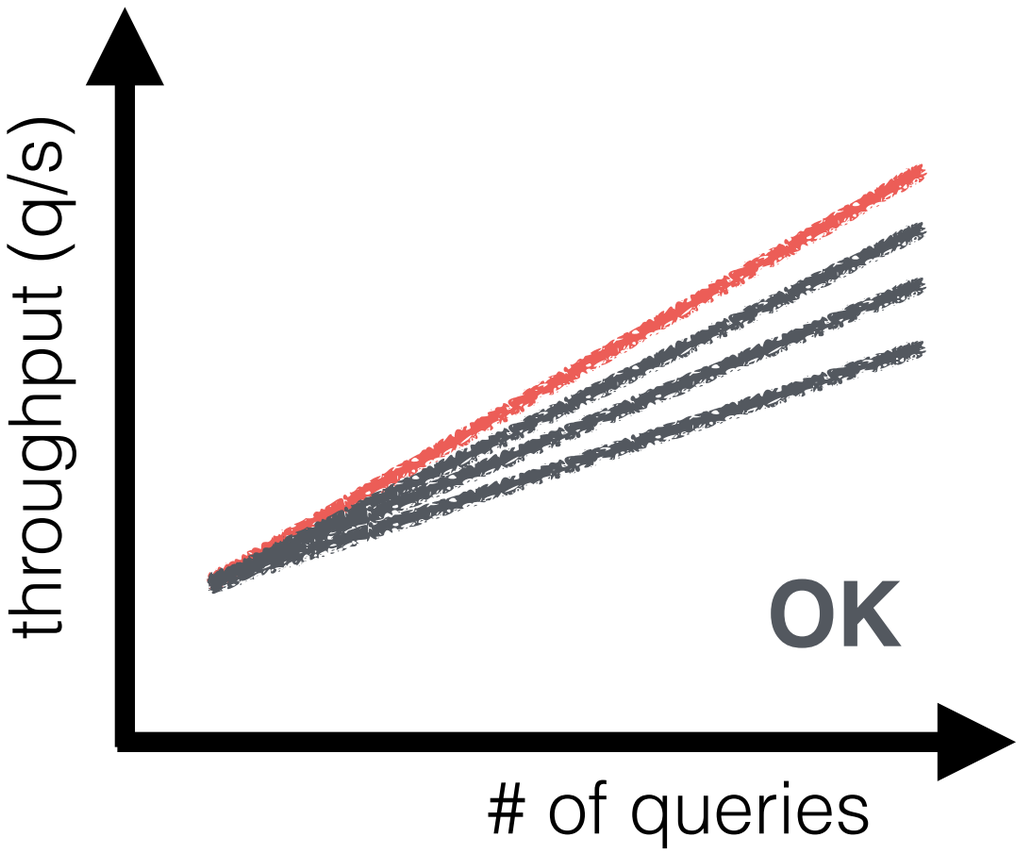
analysis/performance reports

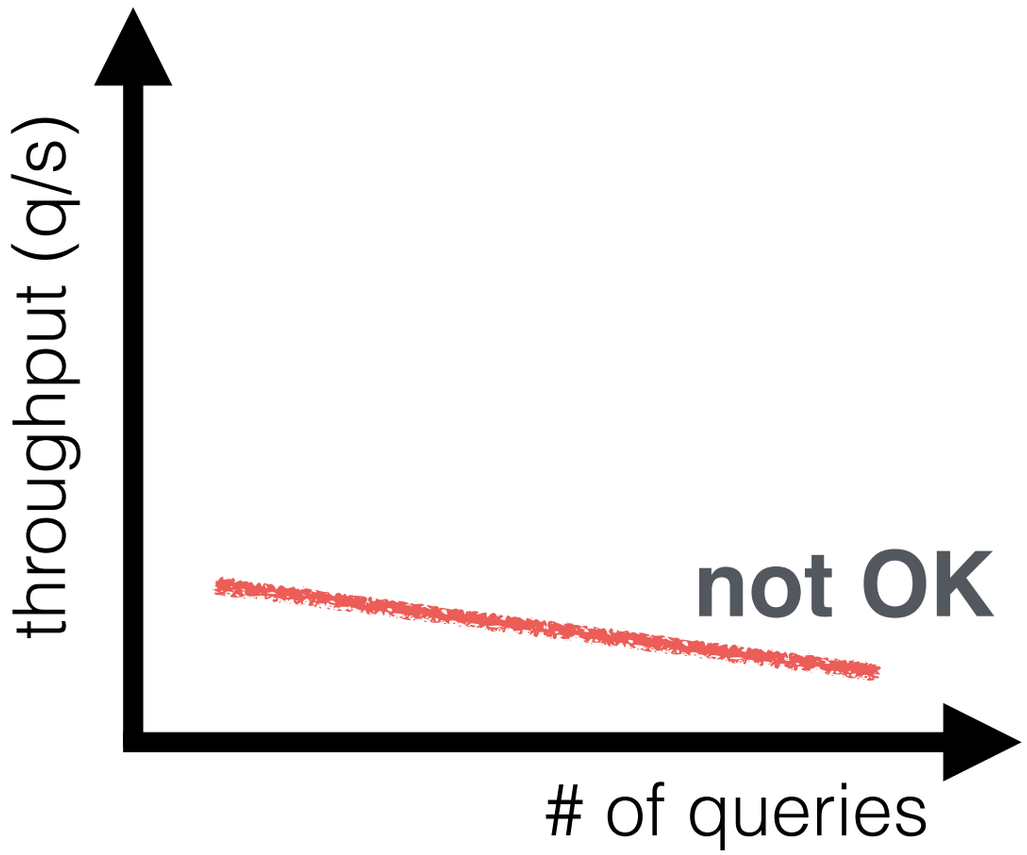




e.g., to test the select operator

analysis/performance reports





e.g., to test shared scans

hardware, data and query based optimizations

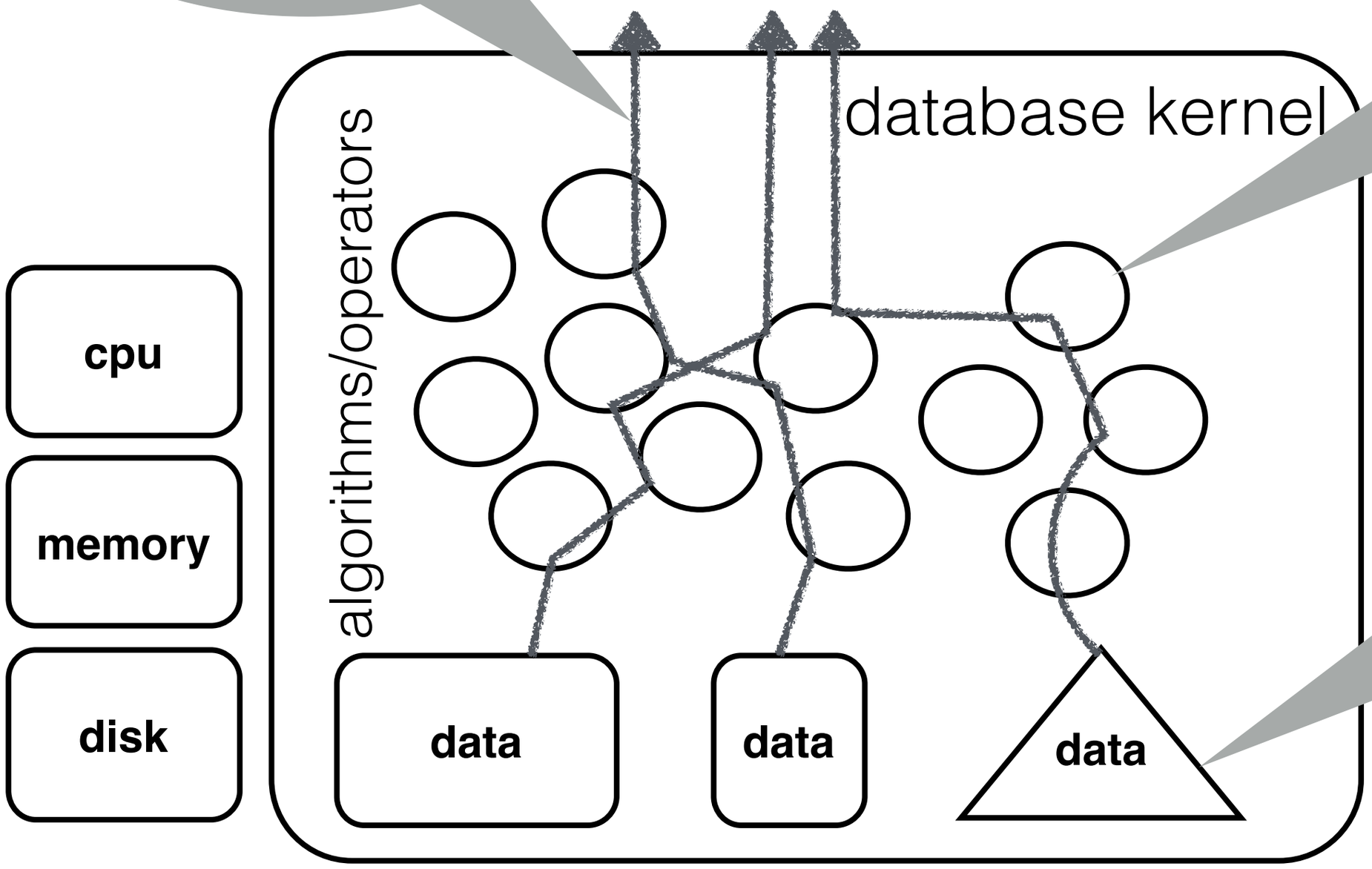
(project=m3)

apply to all
algo/data structures

so far

early/late tuple reconstruction, tuple-at-a-time, vectorized or bulk processing, intermediates format, pushing selects down, etc

scan, binary search, tuple reconstruction, min, max, search/update b-tree, etc.

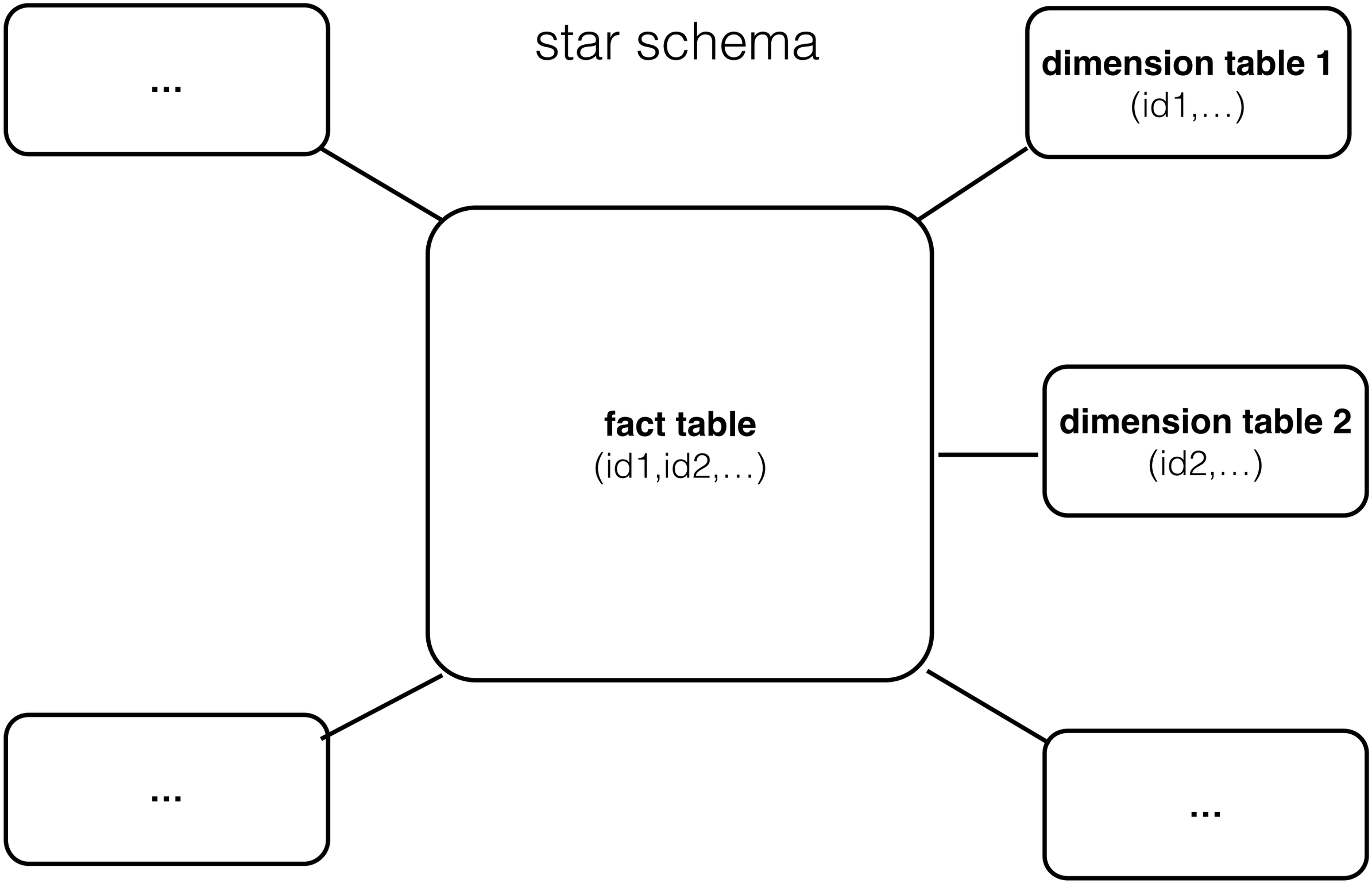


arrays, columns, matrixes, rows,trees

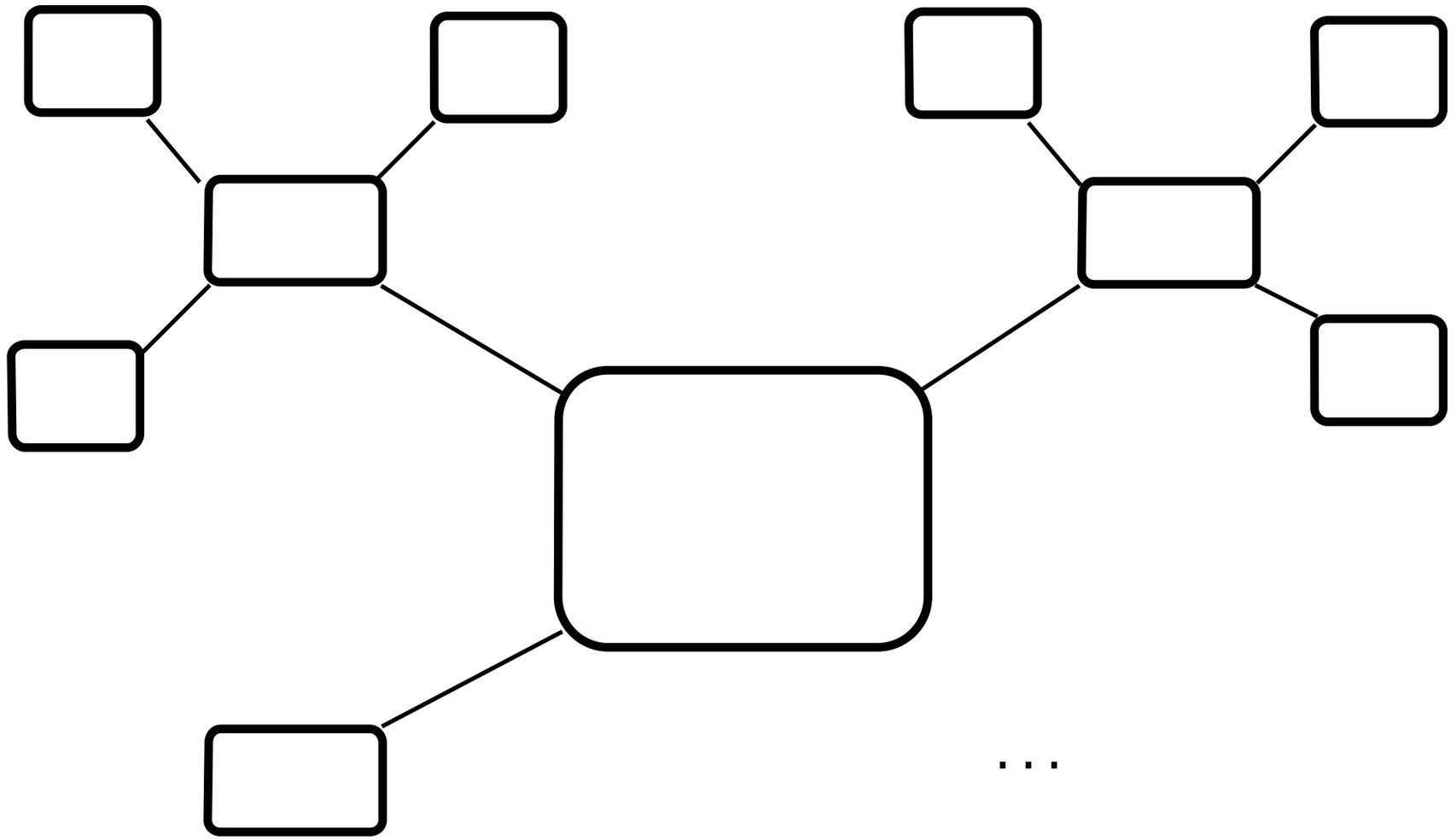
joins

(project=m4)

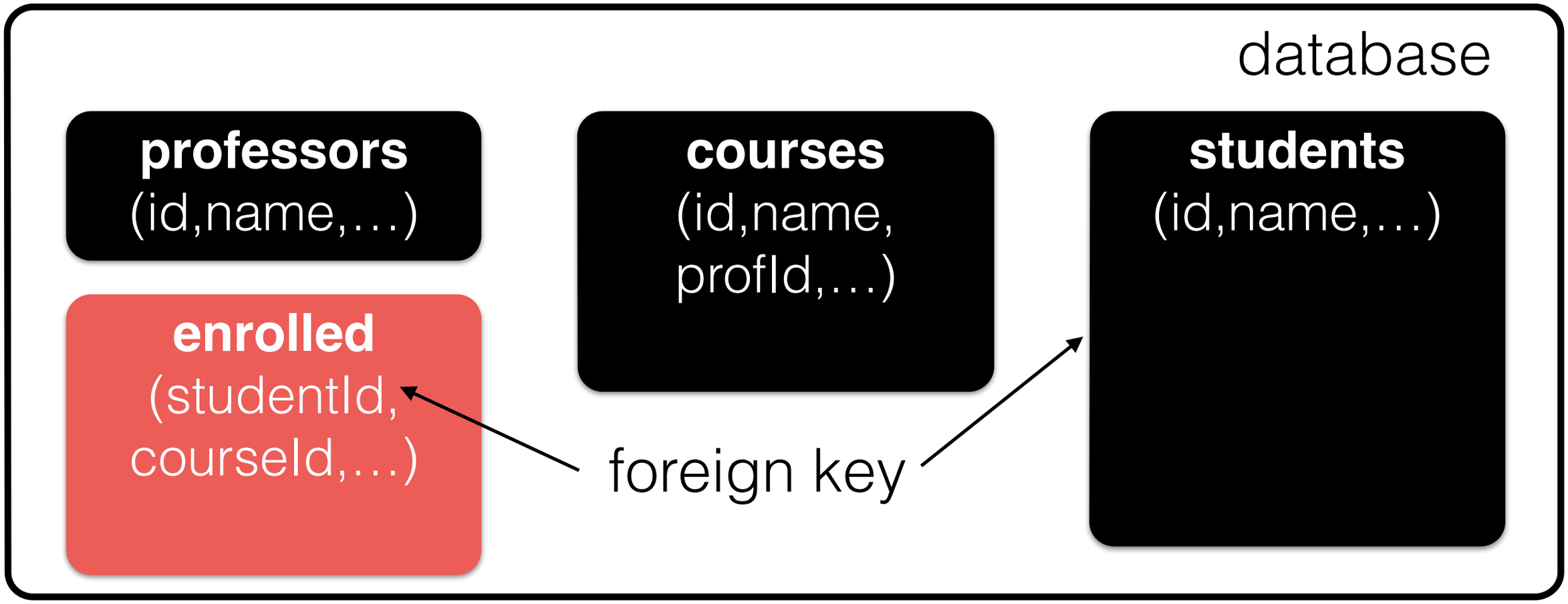
star schema


...
dimension table 1
(id1,…)
fact table
(id1,id2,…)
dimension table 2
(id2,…)
...
...

snowflake schema

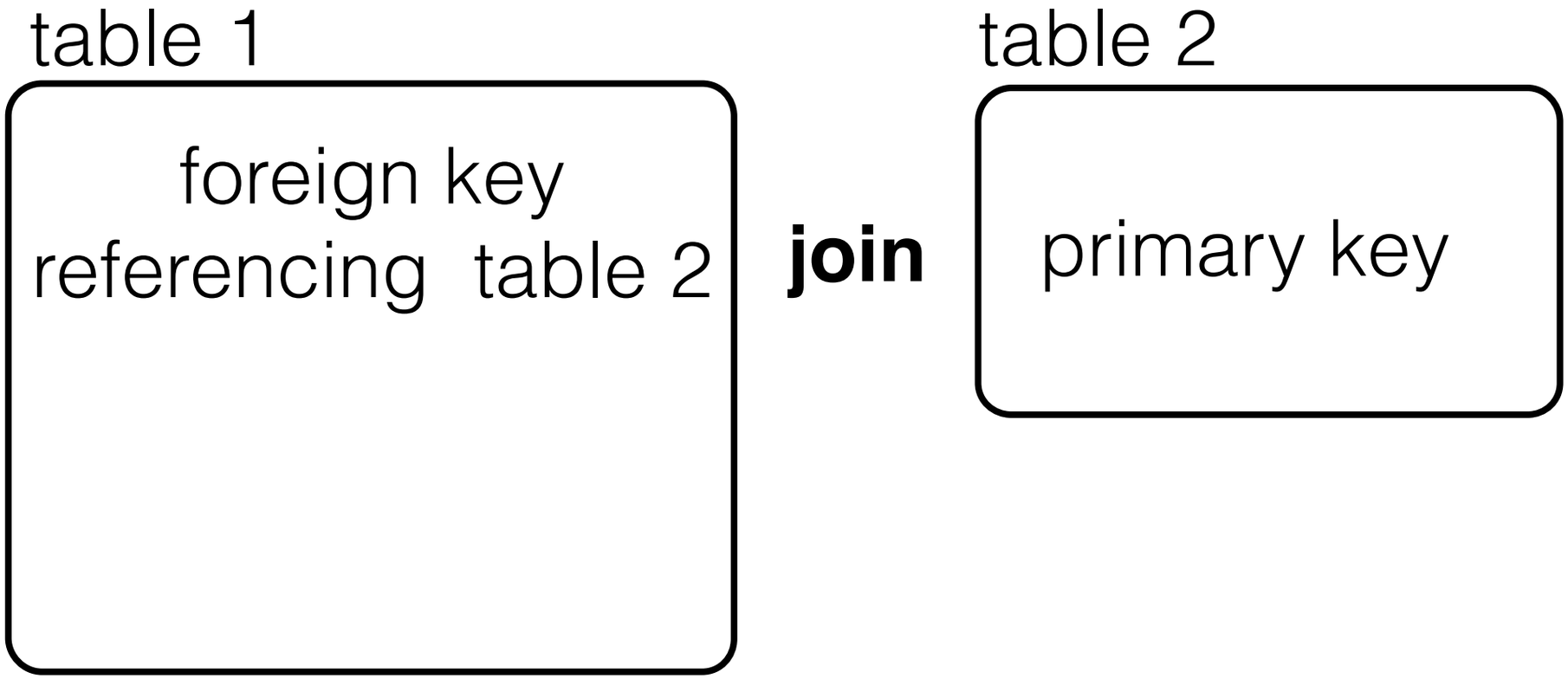


...



give me all students enrolled in cs165

select *student.name* ***from*** *students, enrolled, courses* ***where*** *courses.name="cs165" and enrolled.courseId=course.id and student.id=enrolled.studentId* **join**



find all tuples where FK=PK

join: glue the data back together

key,payload

1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6

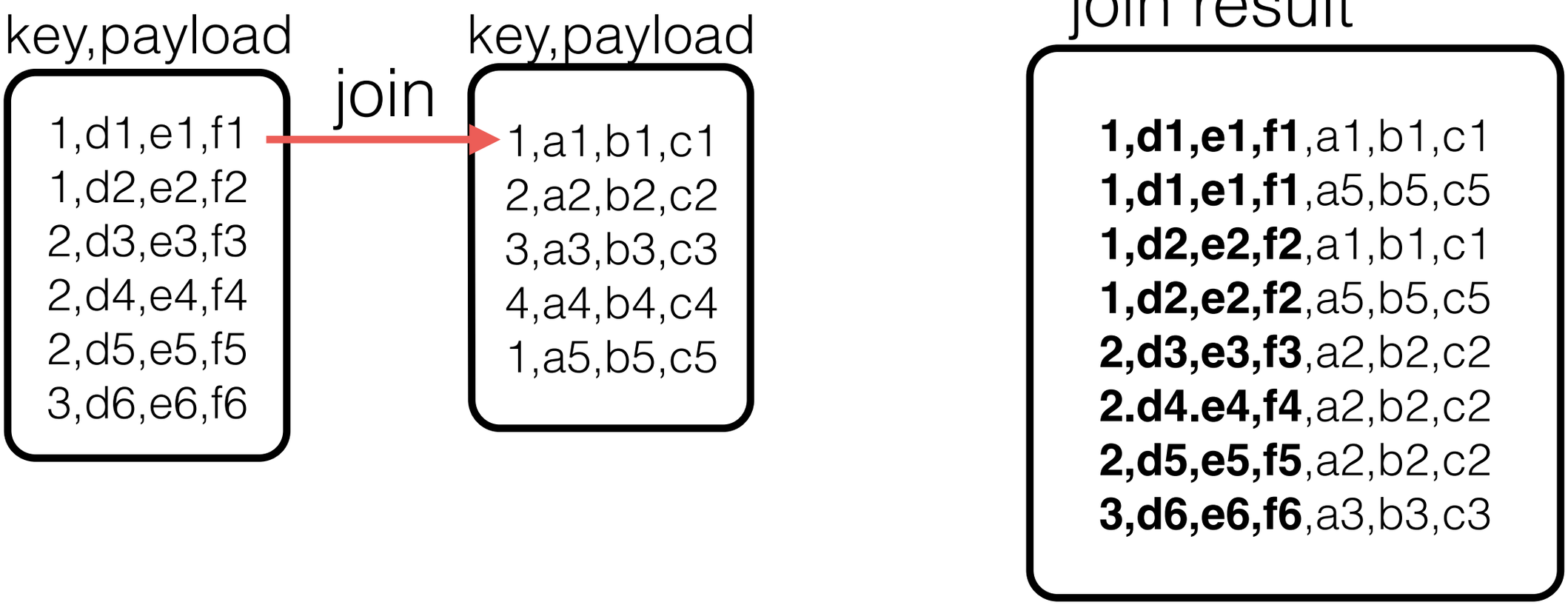
join

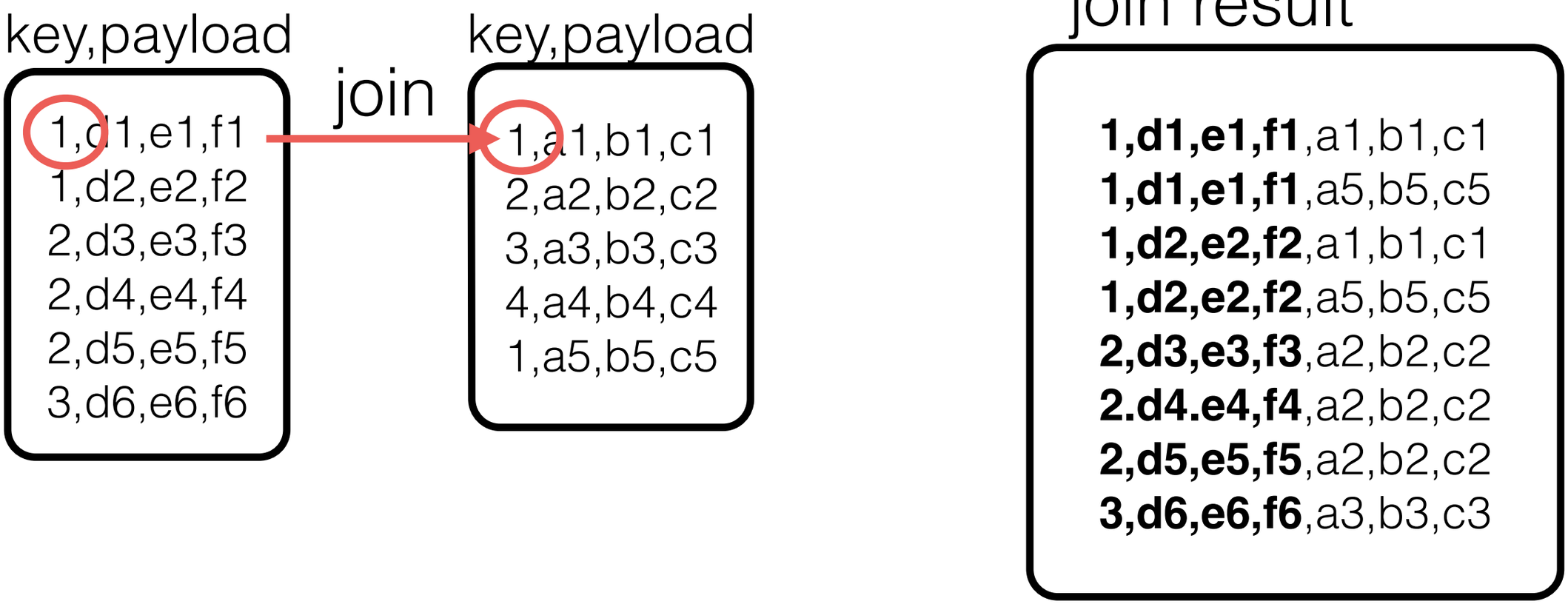
key,payload

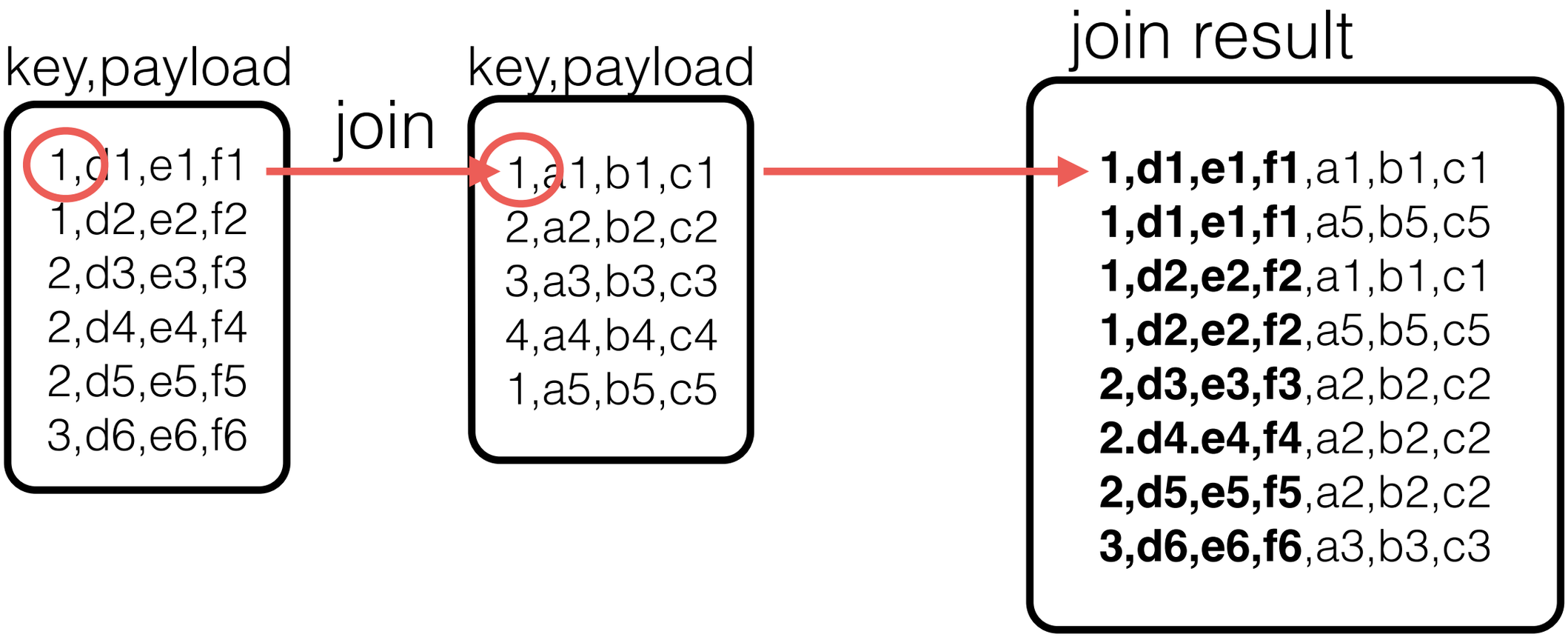
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5

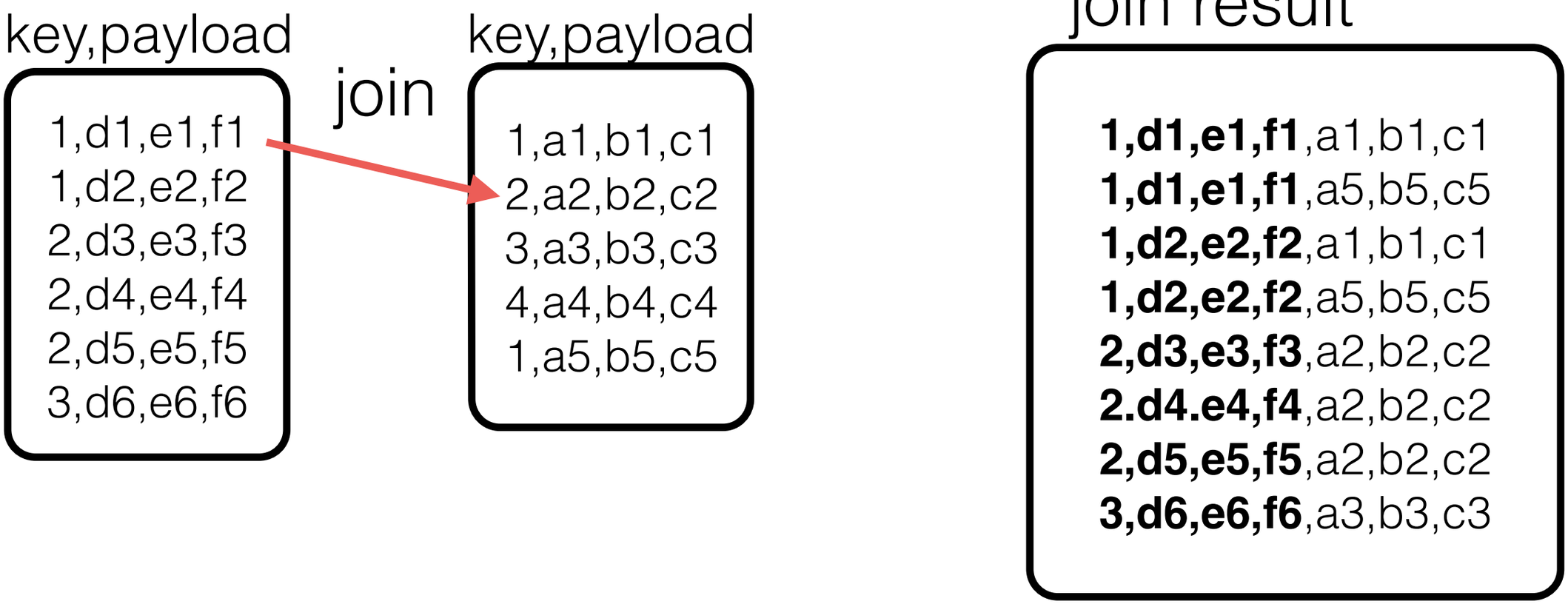
join result

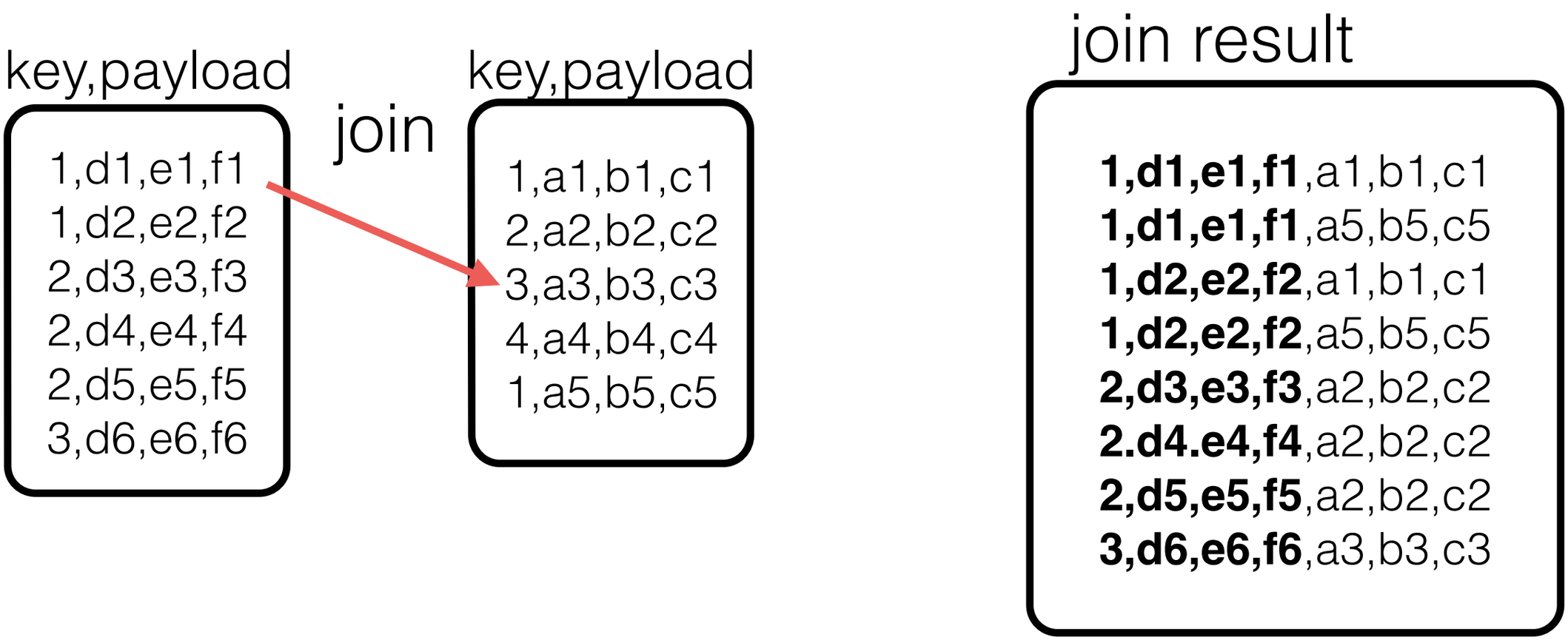
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

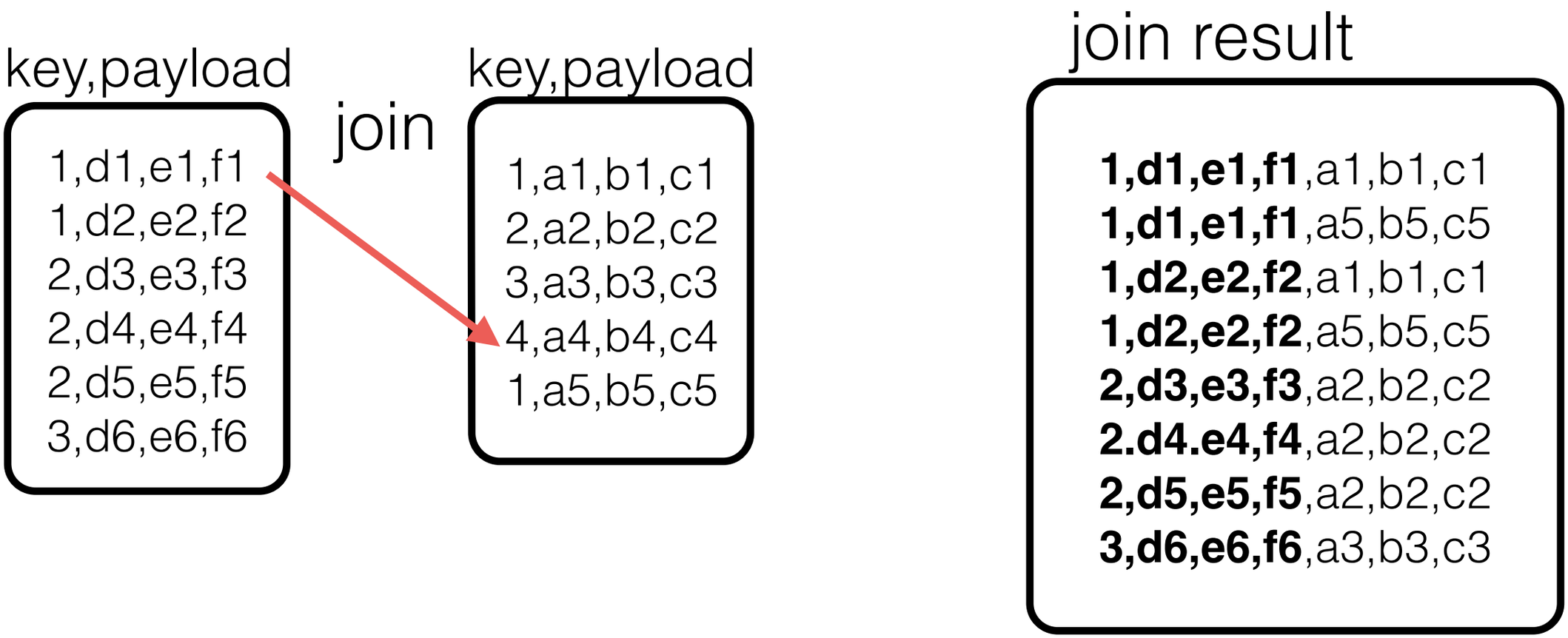

key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

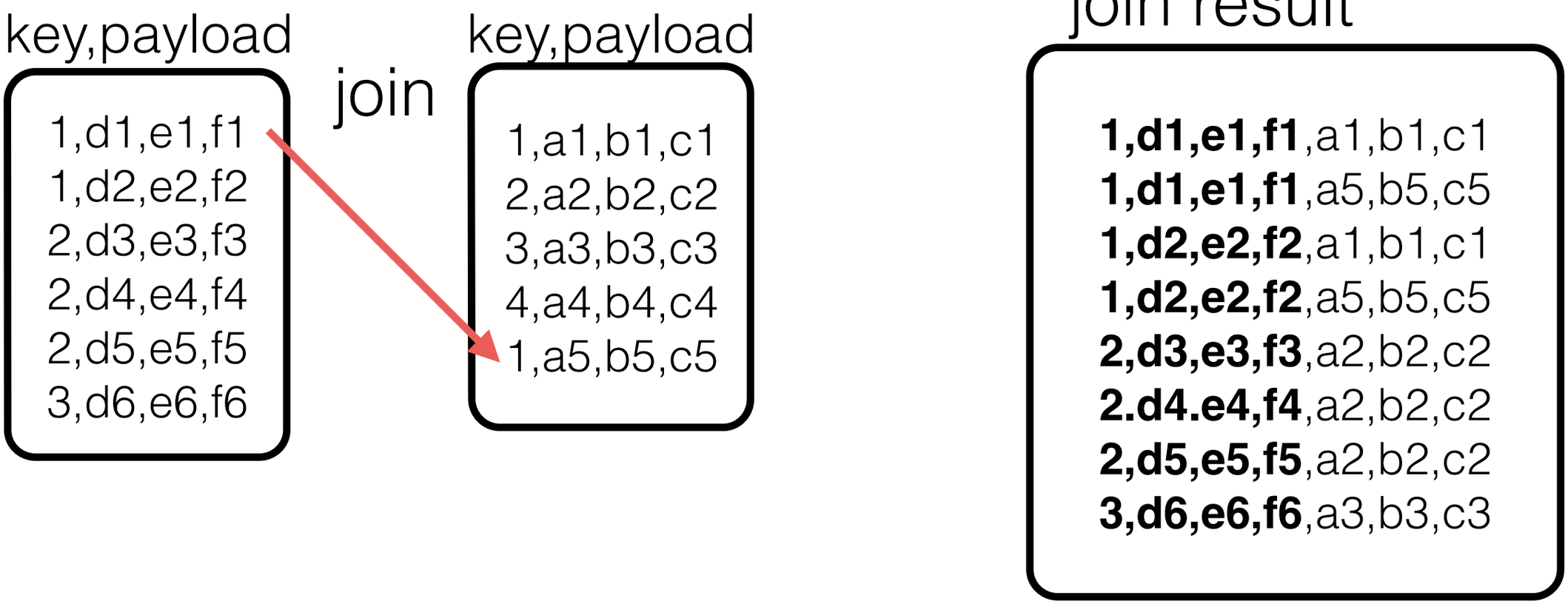

key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

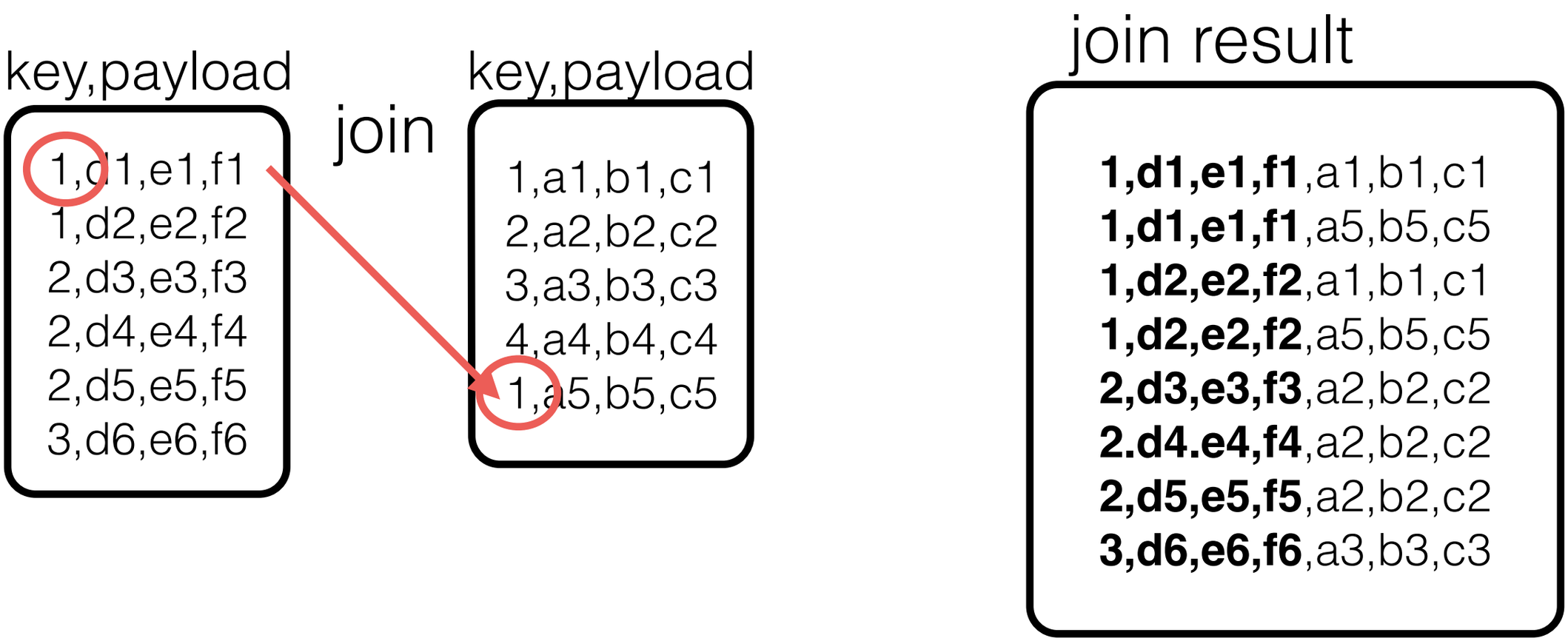

key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

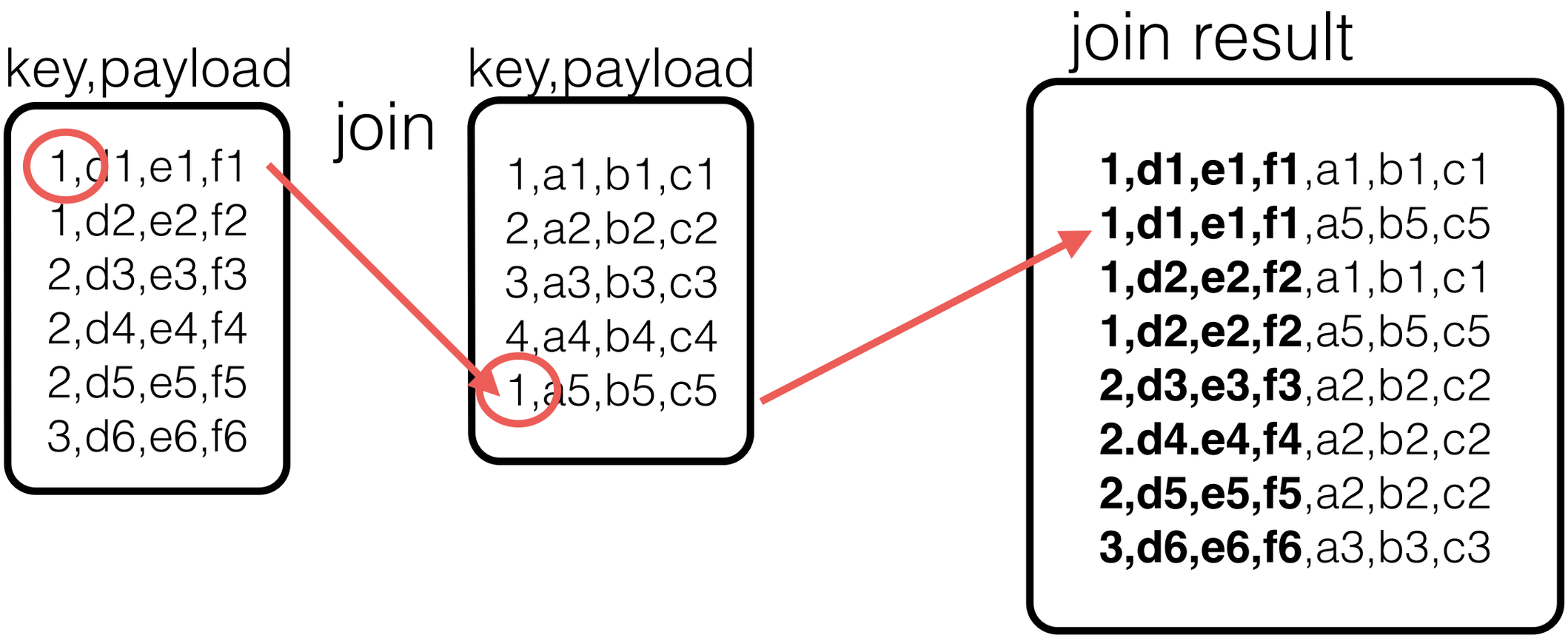

key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3


key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

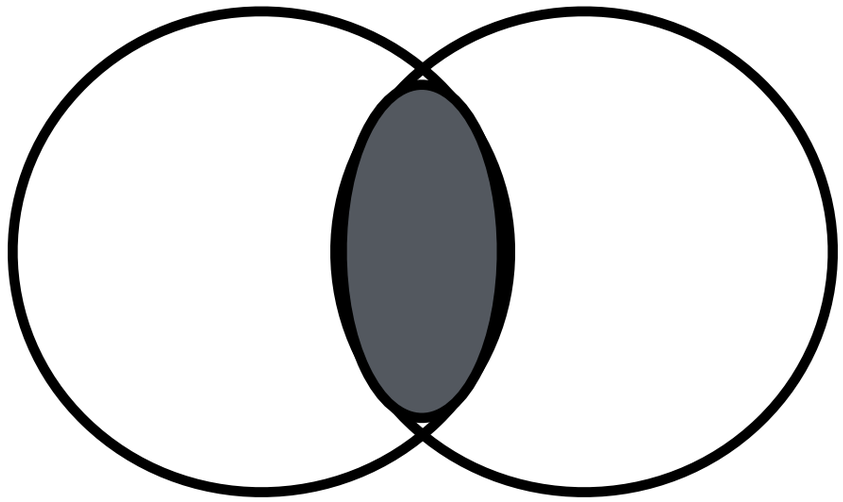

key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3


key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

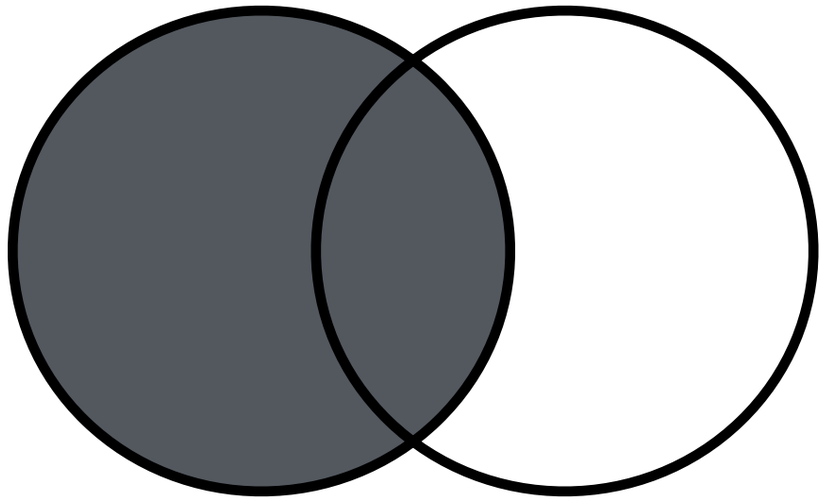

key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3


key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6
join
key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5
join result
1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2.d4.e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3

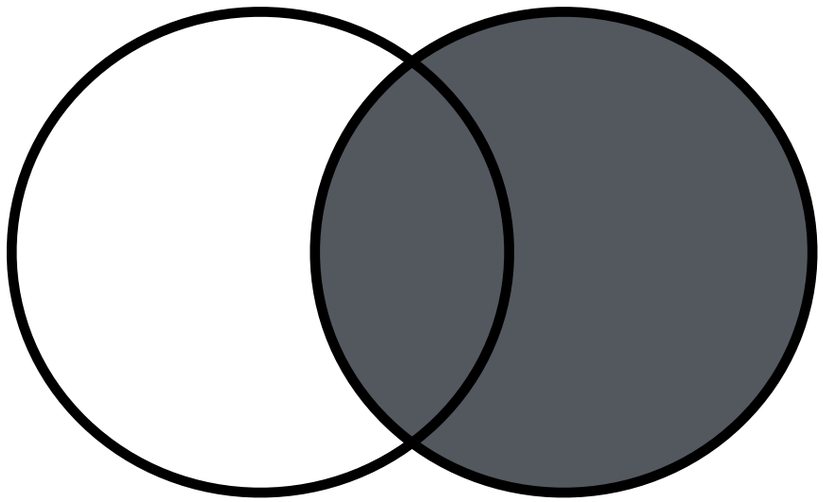
inner join



left outer

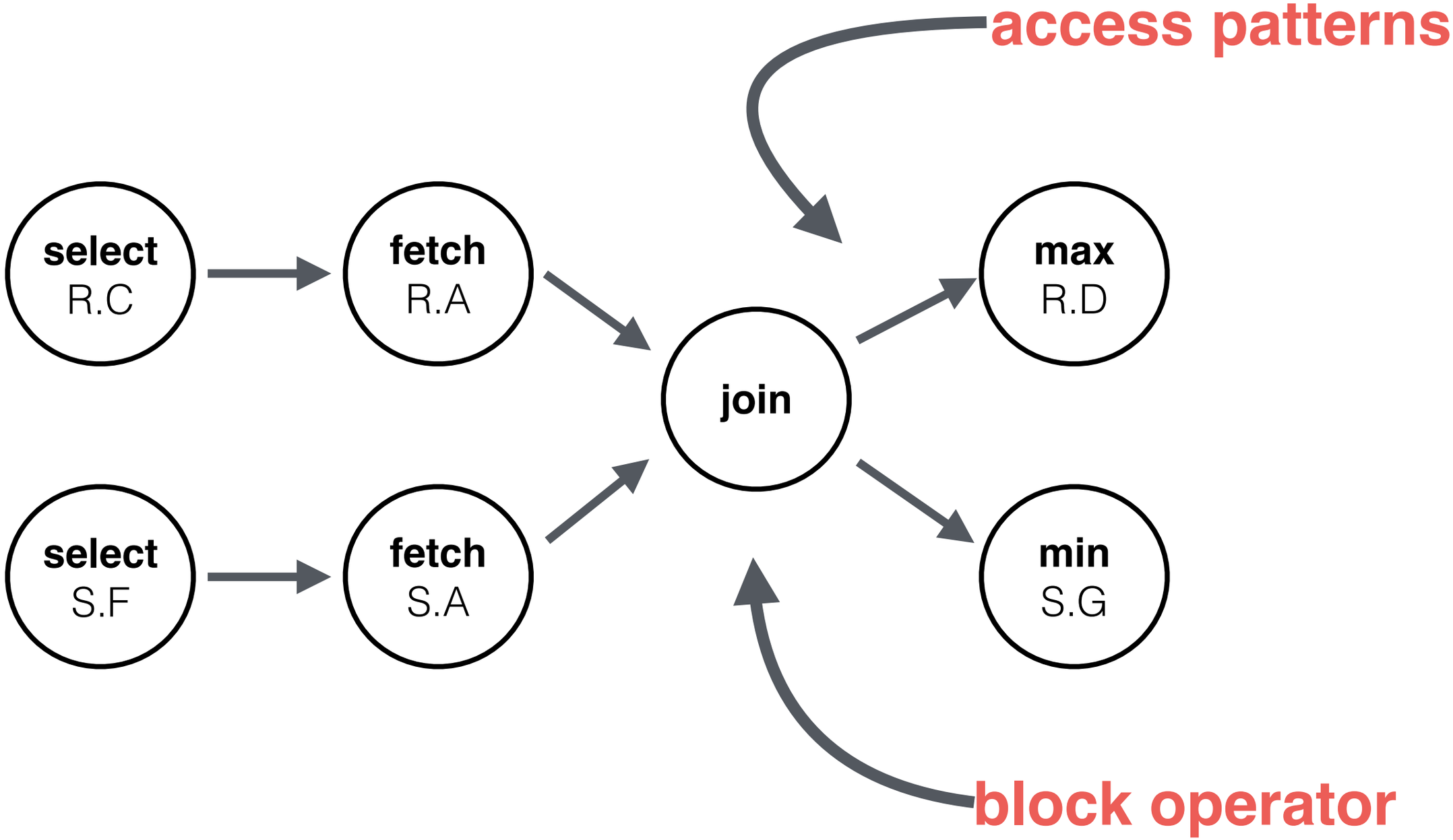


right outer



R(A,B,C,D) - S(A,E,F,G)

select max(R.D),min(S.G)
from R,S
where R.A=S.A and R.C<10 and S.F>30



project
student.name

join
student.id=enrolled.studentid

select
courses.name=“cs165”

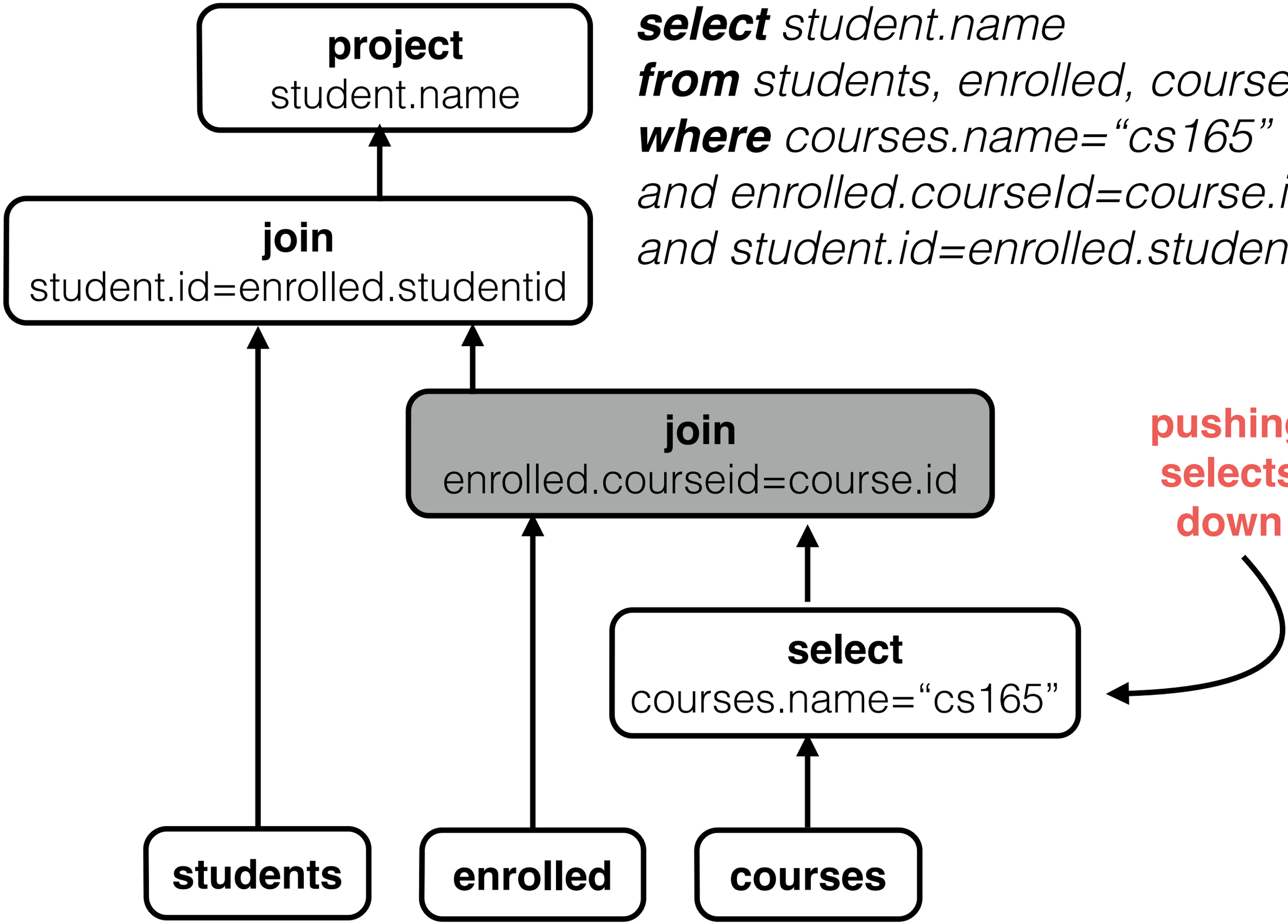
join
enrolled.courseid=course.id

students

enrolled

courses

select *student.name*
from *students, enrolled, courses*
where *courses.name=“cs165”*
and enrolled.courseId=course.id
and student.id=enrolled.studentId



select *student.name*
from *students, enrolled, courses*
where *courses.name=“cs165”*
and enrolled.courseId=course.id
and student.id=enrolled.studentId

pushing selects down

Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 **= select(**Ra,5,20**)**
2. inter2 **= reconstruct(**Rb.inter1**)**
3. inter3 **= select(**inter2,40,50**)**
4. join_input_R **= reconstruct(**Rc,inter3**)**
5. inter4 **= select(**Sa,50,65**)**
6. inter5 **= reconstruct(**Sb,inter4**)**
7. join_input_S **= reverse(**inter5**)**
8. join_res_R_S **= join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)

Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

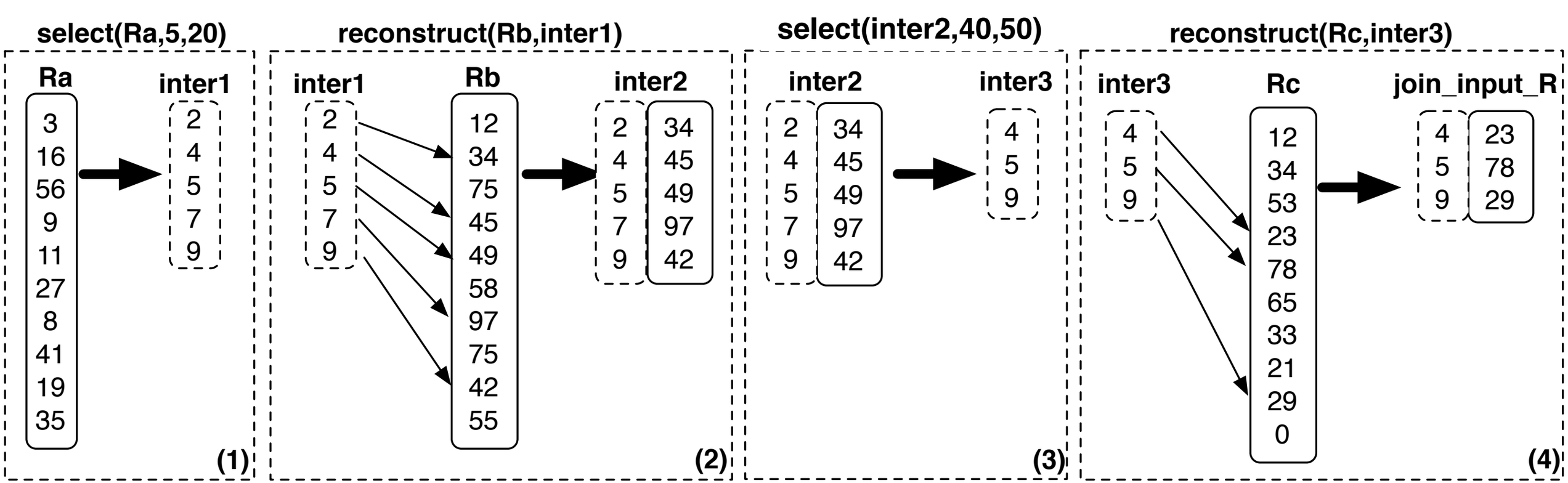
Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = **select(**Ra,5,20**)**
2. inter2 = **reconstruct(**Rb.inter1**)**
3. inter3 = **select(**inter2,40,50**)**
4. join_input_R = **reconstruct(**Rc,inter3**)**
5. inter4 = **select(**Sa,50,65**)**
6. inter5 = **reconstruct(**Sb,inter4**)**
7. join_input_S = **reverse(**inter5**)**
8. join_res_R_S = **join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)



Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 **= select(**Ra,5,20**)**
2. inter2 **= reconstruct(**Rb.inter1**)**
3. inter3 **= select(**inter2,40,50**)**
4. join_input_R **= reconstruct(**Rc,inter3**)**
5. inter4 **= select(**Sa,50,65**)**
6. inter5 **= reconstruct(**Sb,inter4**)**
7. join_input_S **= reverse(**inter5**)**
8. join_res_R_S **= join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)

Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

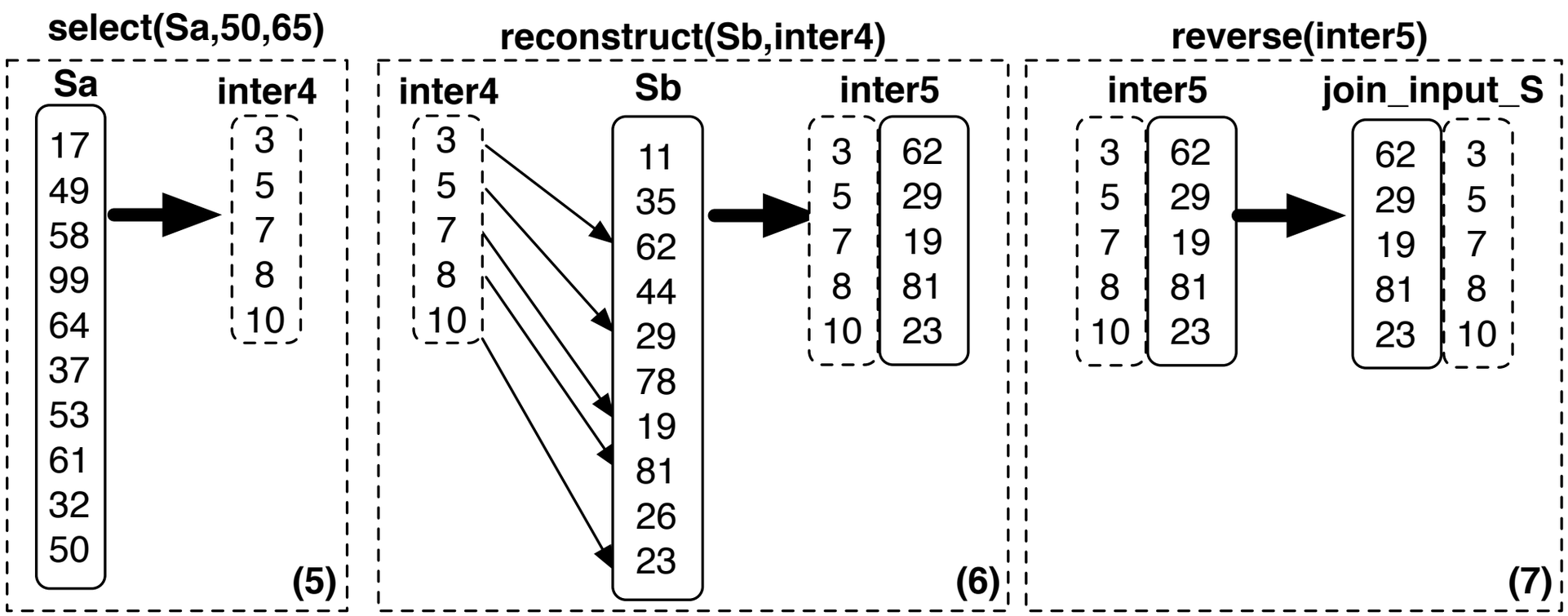
Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = **select(**Ra,5,20**)**
2. inter2 = **reconstruct(**Rb.inter1**)**
3. inter3 = **select(**inter2,40,50**)**
4. join_input_R = **reconstruct(**Rc,inter3**)**
5. inter4 = **select(**Sa,50,65**)**
6. inter5 = **reconstruct(**Sb,inter4**)**
7. join_input_S = **reverse(**inter5**)**
8. join_res_R_S = **join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)



Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 **= select(**Ra,5,20**)**
2. inter2 **= reconstruct(**Rb.inter1**)**
3. inter3 **= select(**inter2,40,50**)**
4. join_input_R **= reconstruct(**Rc,inter3**)**
5. inter4 **= select(**Sa,50,65**)**
6. inter5 **= reconstruct(**Sb,inter4**)**
7. join_input_S **= reverse(**inter5**)**
8. join_res_R_S **= join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)

Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

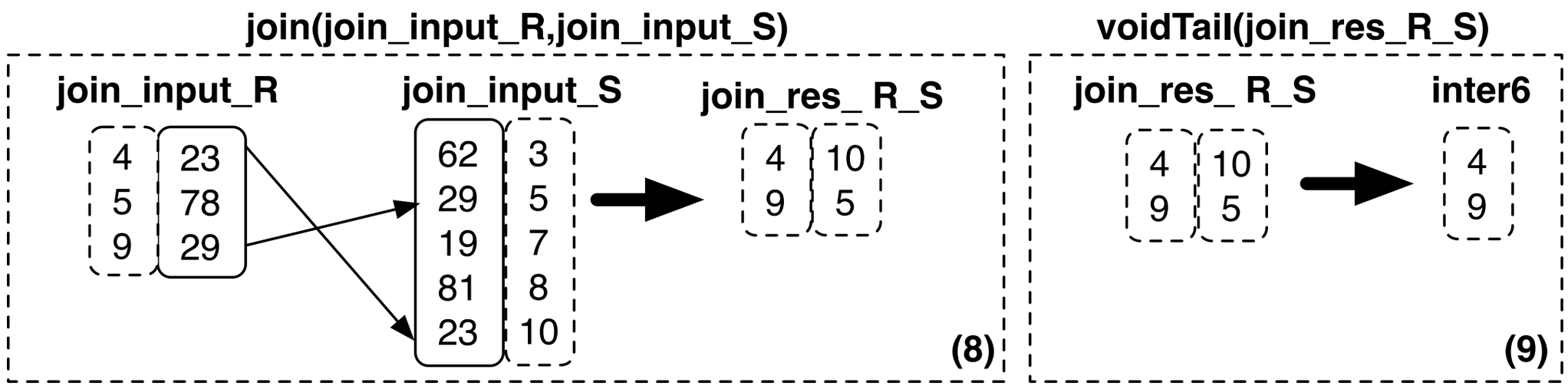
Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = **select(**Ra,5,20**)**
2. inter2 = **reconstruct(**Rb.inter1**)**
3. inter3 = **select(**inter2,40,50**)**
4. join_input_R = **reconstruct(**Rc,inter3**)**
5. inter4 = **select(**Sa,50,65**)**
6. inter5 = **reconstruct(**Sb,inter4**)**
7. join_input_S = **reverse(**inter5**)**
8. join_res_R_S = **join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)



project m4

Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

Relation S

Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 **= select(**Ra,5,20**)**
2. inter2 **= reconstruct(**Rb.inter1**)**
3. inter3 **= select(**inter2,40,50**)**
4. join_input_R **= reconstruct(**Rc,inter3**)**
5. inter4 **= select(**Sa,50,65**)**
6. inter5 **= reconstruct(**Sb,inter4**)**
7. join_input_S **= reverse(**inter5**)**
8. join_res_R_S **= join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)

Initial Status

Relation R

Ra	Rb	Rc
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

Relation S

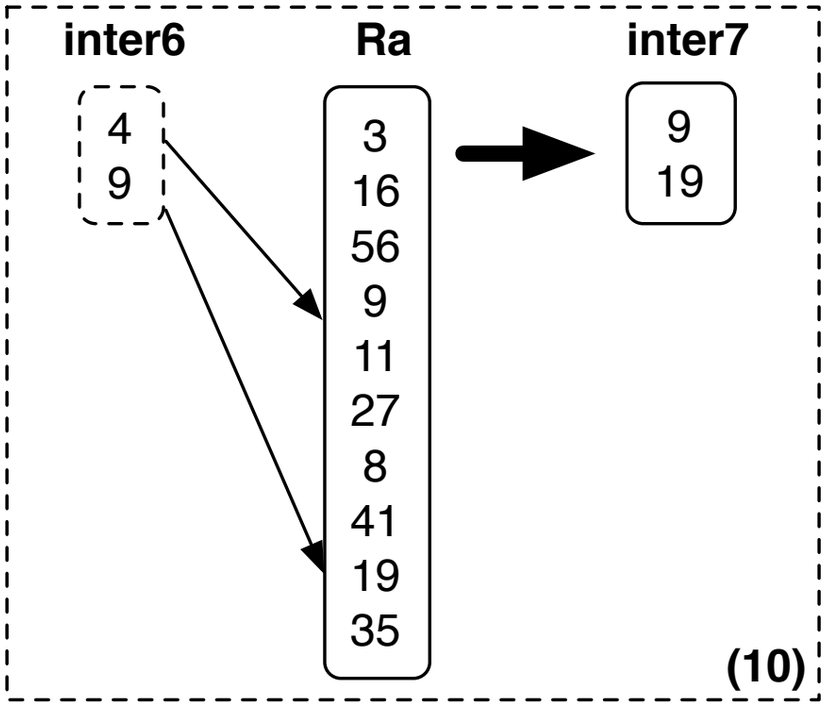
Sa	Sb
17	11
49	35
58	62
99	44
64	29
37	78
53	19
61	81
32	26
50	23

Query and Query Plan (MAL Algebra)

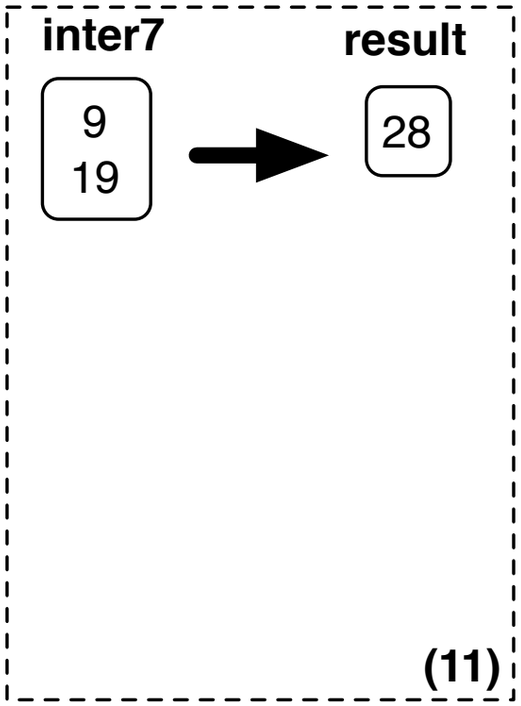
select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = **select(**Ra,5,20**)**
2. inter2 = **reconstruct(**Rb.inter1**)**
3. inter3 = **select(**inter2,40,50**)**
4. join_input_R = **reconstruct(**Rc,inter3**)**
5. inter4 = **select(**Sa,50,65**)**
6. inter5 = **reconstruct(**Sb,inter4**)**
7. join_input_S = **reverse(**inter5**)**
8. join_res_R_S = **join(**join_input_R,join_input_S**)**
9. inter6 = **voidTail(**join_res_R_S**)**
10. inter7 = **reconstruct(**Ra,inter6**)**
11. result = **sum**(inter7)

reconstruct(Ra,inter6)

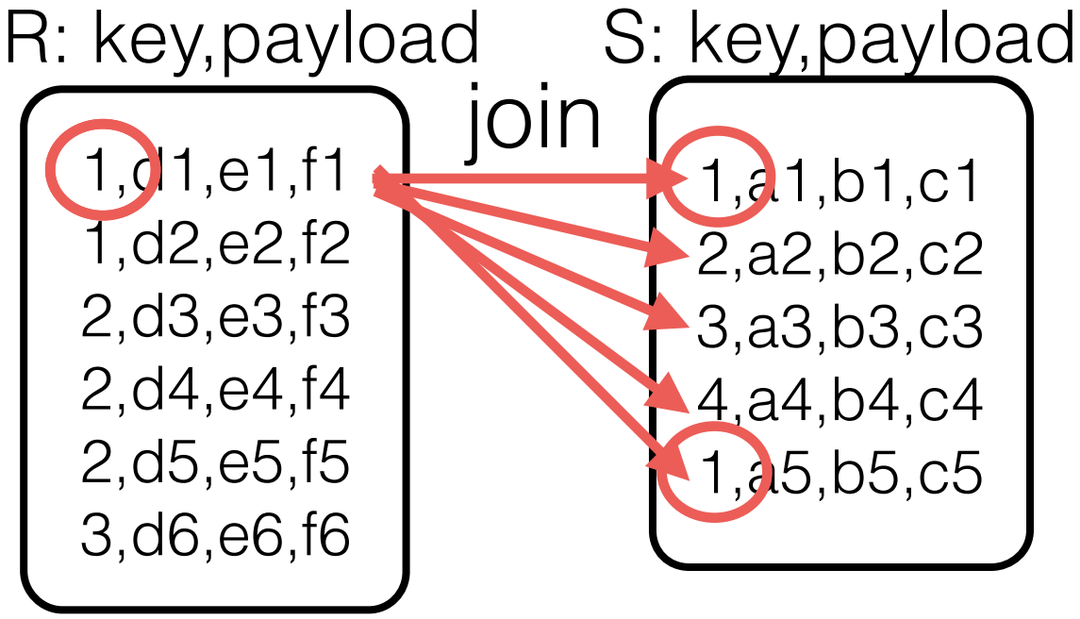


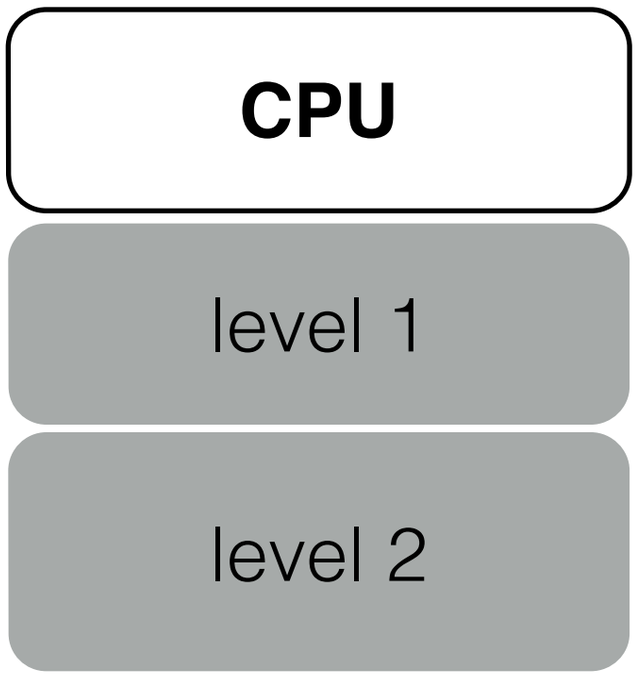
sum(inter7)



for every
L.key = R.key pair
return [L.pos,R.pos]

data/results
one column-at-a-time



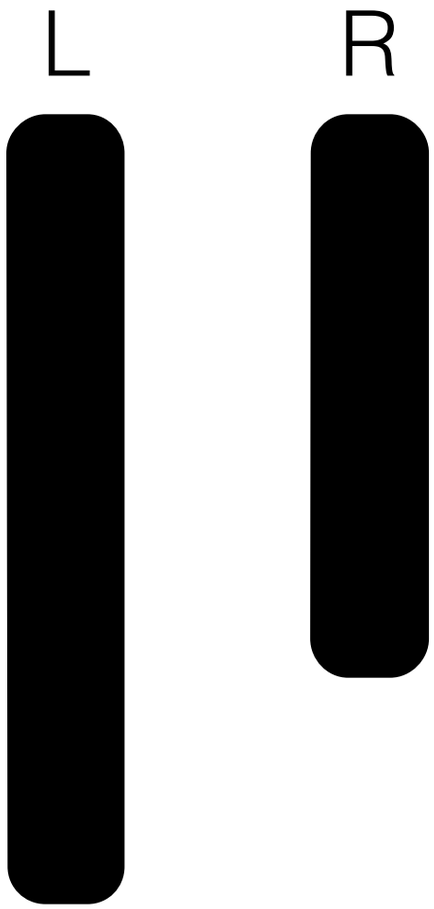


1) design a nested loops join algorithm and give its cost
2) which table do we start from?
3) think of optimizations to minimize L1 misses
4) can we use sorting?

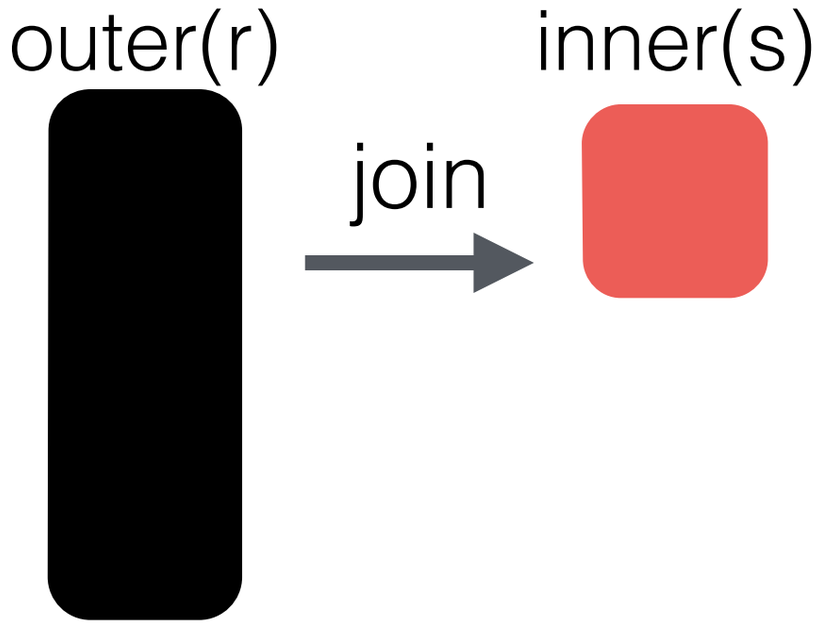
Assume we have a b-tree on one of the join columns:
5) how would you use the b-tree to do a join?
6) would you use the b-tree or the nested loops join?

all R columns > L1, all S columns >L1, R+S << L2, L1 block = L2 block
CPU can read directly from L1 only

nested loops



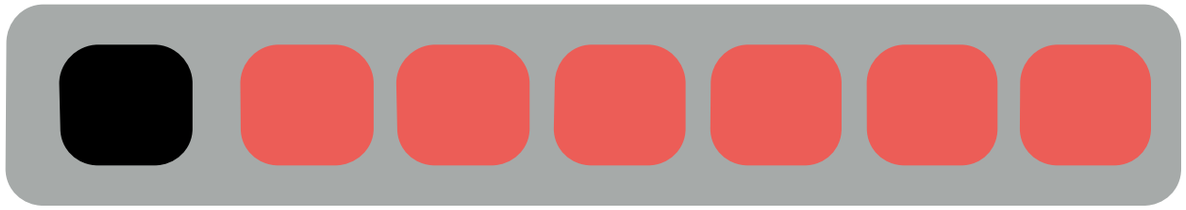

```
new resL[]; new resR[]; k=0
for (i=0;i<L.size;i=i++)
    for (j=0;j<R.size;j++)
        if L[i]==R[j]
            resL[k]=i
            resR[k++]=j
```



probe the small side
for (possibly) better **data locality**

r/p+s/p, s<L1 Level1

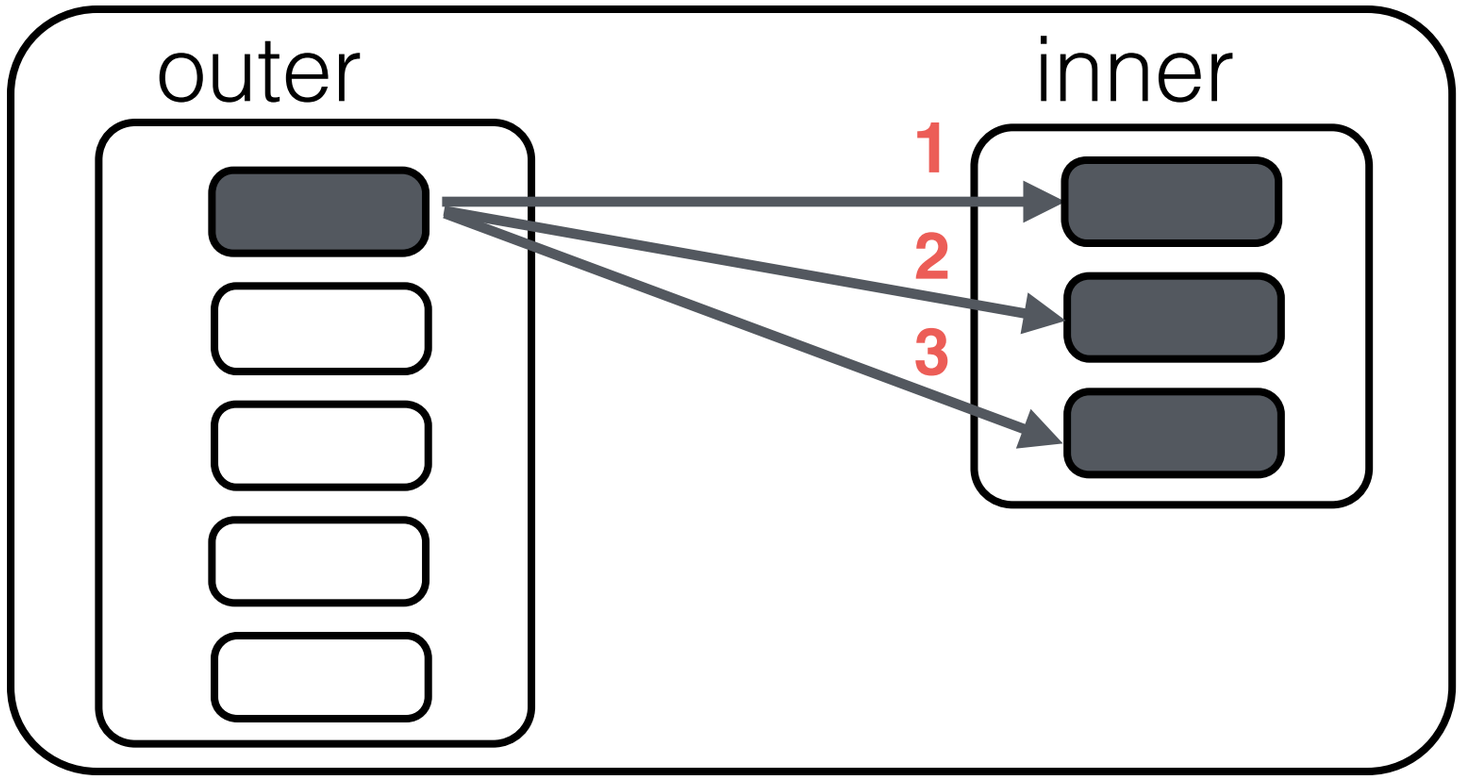
stream outer pages
(try to) hold inner pages



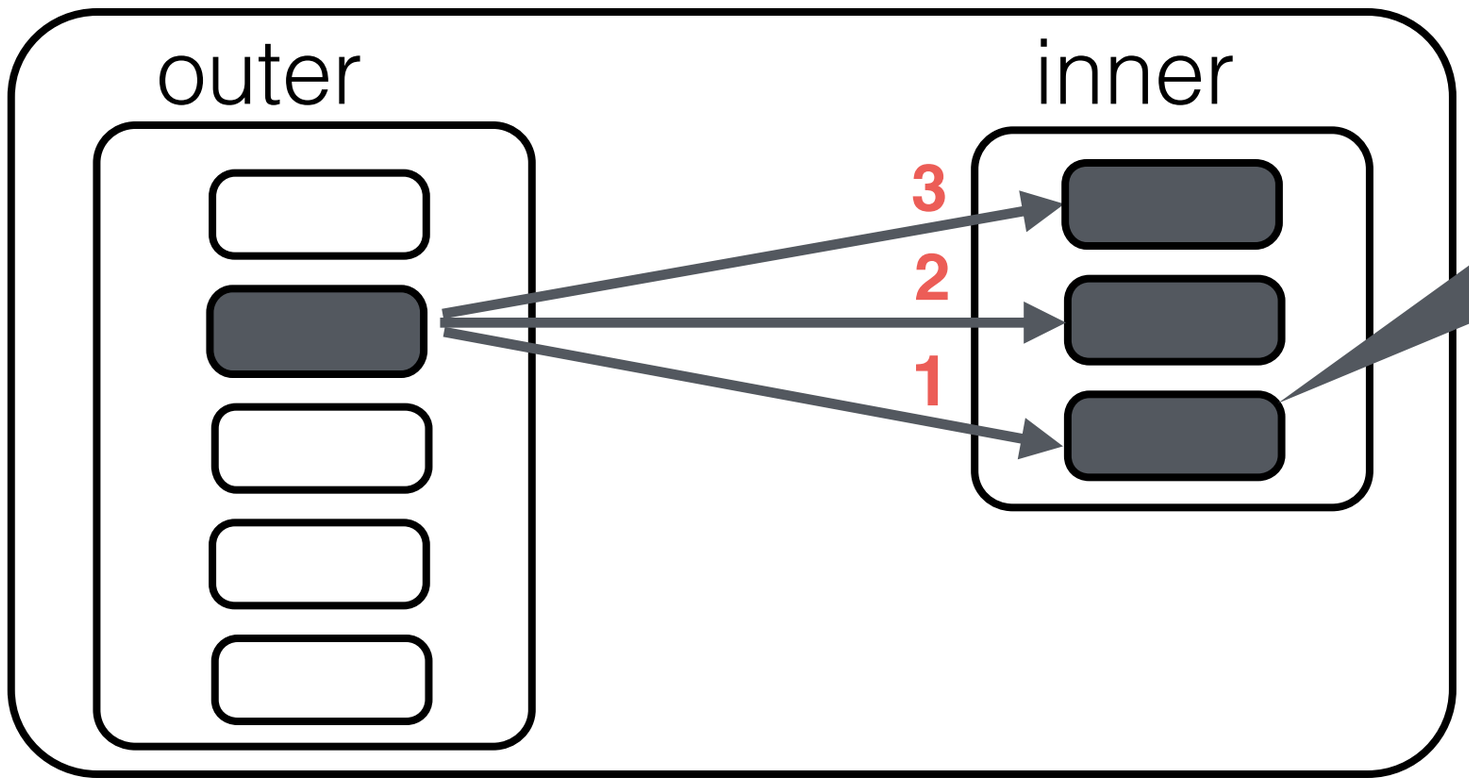
Level2

zig zag

phase 1



phase 2



will still be in LLC

...

blocked nested loops

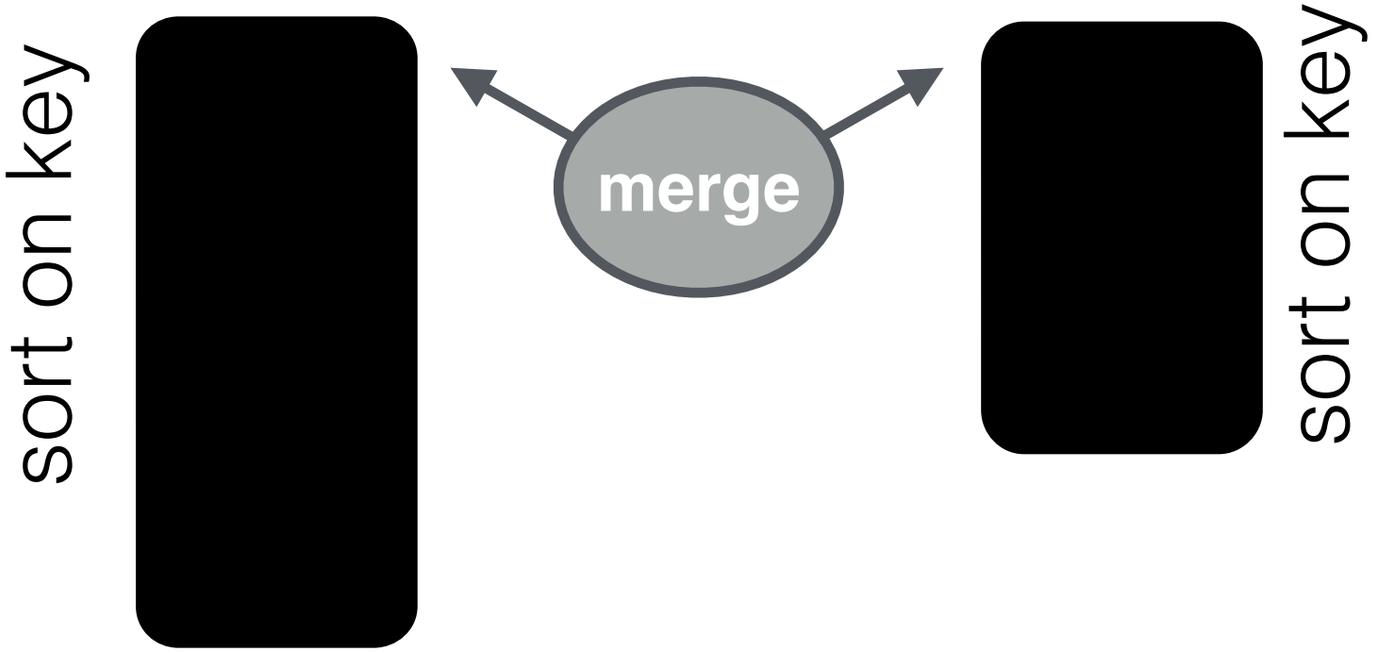
```
new resL[]; new resR[]; k=0
for (i=0;i<L.size;i=i++)
     for (j=0;j<R.size;j++)
          if L[i]==R[j]
               resL[k]=i
               resR[k++]=j
```

comp *O(LxR)*
I/O *O(L/p+Lx(R/p))*

```
new resL[]; new resR[]; k=0
for (i=0;i<L.size;i=i+pagesize)
     for (j=0;j<R.size;j=j+pagesize)
          for (r=i;r<i+pagesize;r++)
               for (m=j;m<j+pagesize;m++)
                    if L[r]==R[m]
                         resL[k]=r
                         resR[k++]=m
```

comp *O(LxR)*
I/O *O(L/p+(L/p)x(R/p))*

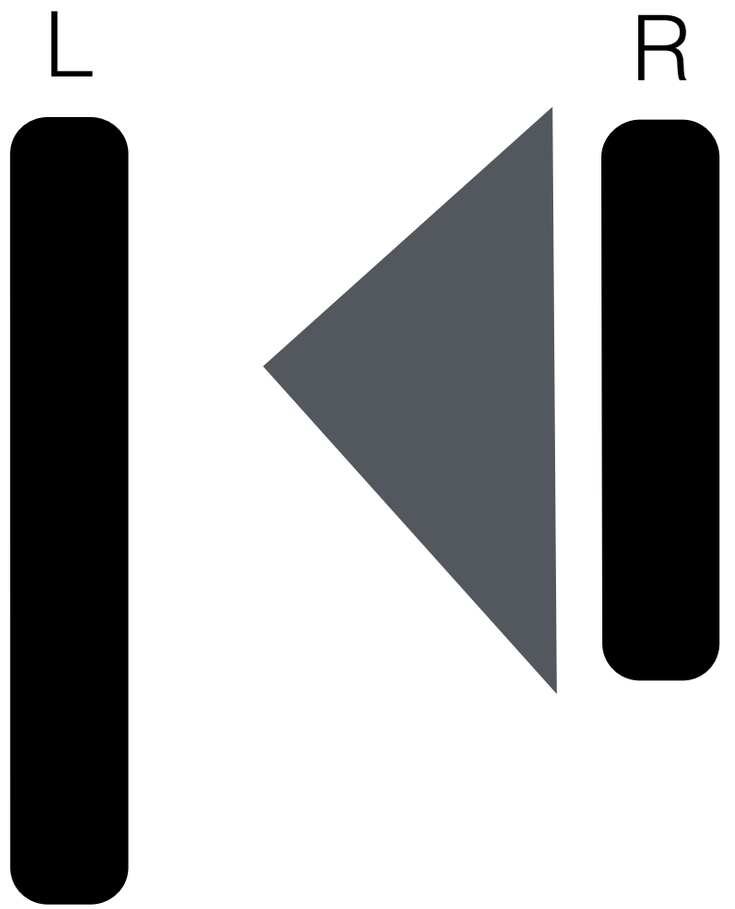
sort merge join



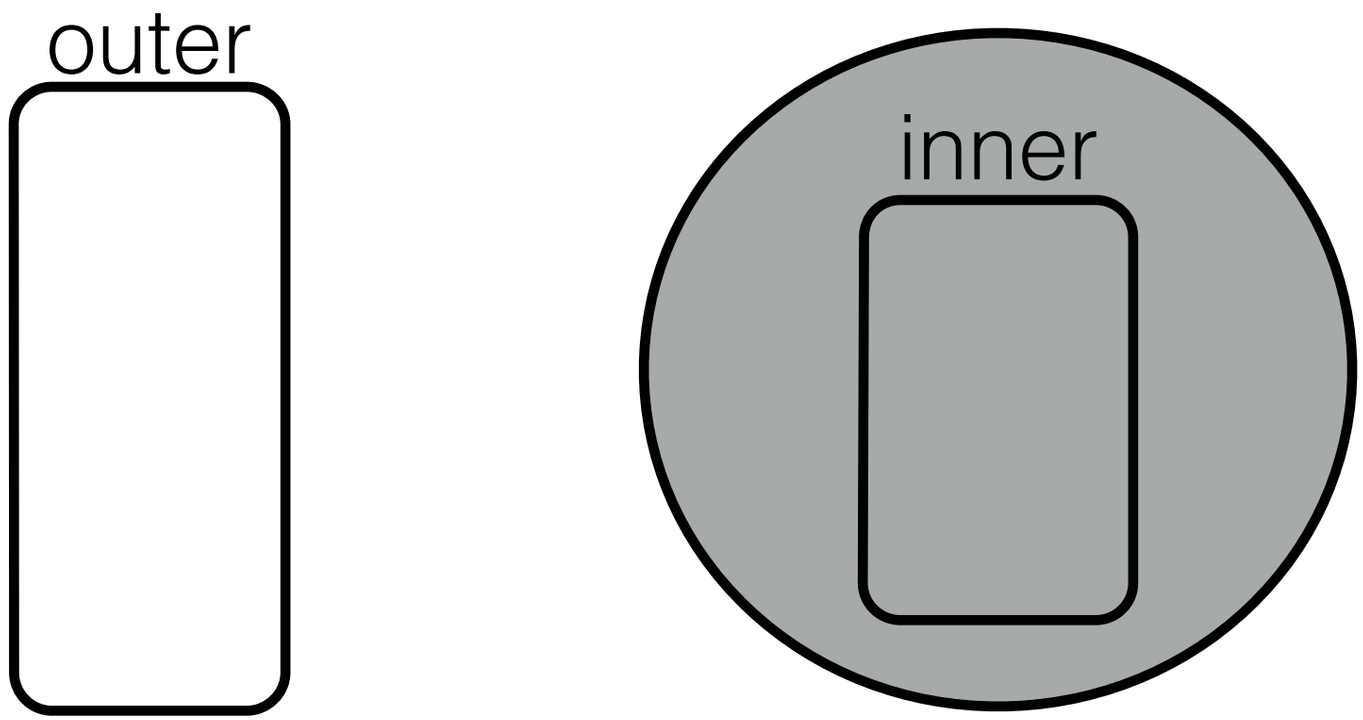
perfect if data is sorted or needs to be sorted anyway for the rest of the plan

```
while left and right still have tuples
    if left.val < right.val left++
    else if left.val > right.val right++
    else
        add to result, left++, right++
```

```
new resL[]; new resR[]; k=0
for (i=0;i<L.size;i=i++)
      jk=R.btree.probe(L[i])
      if (jk!=null)
            resL[k]=i
            resR[k++]=jk.pos
```



even with buffering, we might have to read
several (random) pages per probe (to traverse the tree)



oracle

search in O(1)

what if not all data fits in **main-memory**?

what if not all data fits in **L3 cache**?

what if not all data fits in **L2 cache**?

what if not all data fits in **L1 cache**?

can we exploit the **>1 cores** in modern **CPUs**?

joins

(project=m4)

what to do in m4?

nested loops and hash joins

cache conscious and multi-core (within reason)

textbook: chapters 4, 14

joins

DATA SYSTEMS

prof. Stratos Idreos