Page 1

```java
public static String kangaroo(int x1, int v1, int x2, int v2) {

    if (v1 > v2) {
        int count = 1;
        while ((x1 + v1 * count) <= (x2 + v1 * count)) {
            if (x1 + v1 * count == x2 + v2 * count)
                return "YES";
            count++;
        }
    }
    return "NO";
}
```

for me this is UNCLEAR

you spin CPU enormously

$S = v \cdot t$

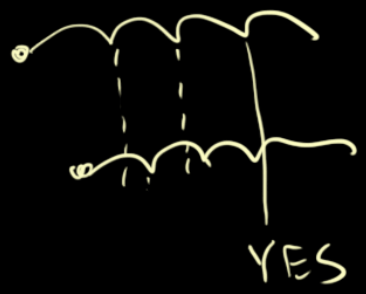
number of jumps

velocity

distance

Iterative

Math




```java
String word = "NO";
    while (x1 < x2) {
        if ((x2 > x1 && v2 > v1) || (x1 > x2 && v1 > v2)) {
            break;
        }
        x1 = x1 + v1;
        x2 = x2 + v2;
        if (x1 == x2) {
            word = "YES";
        }
    }
    return word;
}
```

Page 2

```
static String kangaroo(int x1, int v1, int x2, int v2) {

    if (x1+v1==x2+v2) return "YES";
    if (x1>x2 && v1>v2 || x2>x1 && v2>v1) return "NO";
    if (x1>x2 && v1==v2 || x2>x1 && v1==v2) return "NO";
    else return kangaroo(x1+v1,v1,x2+v2,v2);

}
```

it still iterative approach

(v)

```
static String kangaroo(int x1, int v1, int x2, int v2) {

        return v1-v2==0 || (x2-x1)/(v1-v2)<0 || (x2-x1)%(v1-v2)!=0?"NO":"YES";
    }
```

(v)

```
return (v2 - v1 == 0 || (x2 - x1) / (v1 - v2) < 0) ? "NO" :
       (x2 - x1) % (v1 - v2) == 0 ? "YES" : "NO";
```

(v)

Page 3

```
static String kangaroo(int x1, int v1, int x2, int v2) {
    if(v1>v2 && (x2-x1)%(v1-v2)==0){
        return "YES";
    }else {
        return "NO";
    }

}
```

← put it to ?:

```
static String kangaroo(int x1, int v1, int x2, int v2) {
    if(v1>v2 && (x2-x1)%(v1-v2)==0){
        return "YES";
    }else {
        return "NO";
    }

}
```

```
static String kangaroo(int x1, int v1, int x2, int v2) {
    return x2>x1&&v2>=v1 ? "NO" : checkjump(x1,v1,x2,v2) ?"YES" :"NO";
}
private static boolean checkjump(int x1, int v1, int x2, int
    return (x1-x2)%(v2-v1)==0;
}
```

```
static String kangaroo(int x1, int v1, int x2, int v2) {
    int distance = x1 - x2;
    int velocity = v1 - v2;
    return (velocity * distance < 0) && (distance % velocity == 0)?"YES":"NO";
}
```

① . I want to check whether these K can meet

② . solve it

③ . I want to represent result in the separate fn.

f(x1, x2, v1, v2) → Boolean

Page 5

```
static int countingValleys(int n, String s) {
    int valleys = 0;
    int level = 0;
    for (int i = 0; i < s.length(); i++) {
        if(s.charAt(i)=='U'){
            level++;
        }
        else{
            level--;
            if(level == -1){
                valleys++;
            }
        }
    }
return valleys;
    }
```

'D'?

iteration

too complicated

- because everything in one place
- extremely big level of nesting

you write the code for yourself tomorrow

it is extremely easy to introduce a mistake

Page 6

```
static int countingValleys(int n, String s) {
        int plus=0;
        int minus=0;

        for(int i=0; i<n; i++){
            char ch=s.charAt(i);
            String road=Character.toString(ch);
            if(road.equals("D")){
                if(minus==0){
                    plus+=1;
                }
                minus-=1;
            }
            else {
                minus+=1;
            }

        }
return plus;
    }
```

```
static int countingValleys(int n, String s) {

        int count = 0;
        int ds = 0;

        for (int i = 0; i < n; i++) {

            switch (s.charAt(i)) {
                case 'U':
                    if (ds == -1)  count++; ds++; break;
                case 'D':
                    ds--;
            }

        }
        return count;

    }
```

Way better

Page 7

```java
static int countingValleys(int n, String s) {
    int count = 0;
    int track = 0;
    for (int i = 0; i < s.length(); i++) {
        track += s.charAt(i)=='U' ? 1 : -1;
        count += track==0 && s.charAt(i)=='U' ? 1 : 0;
    }
    return count;
}
```

```java
static int countingValleys(String s) {
    int valleys = 0;
    int steps = 0;
    for (int i = 0; i < s.length(); i++) {
        switch (s.charAt(i)) {
            case 'U': steps++; break;
            case 'D': steps--; break;
        }
        if (steps == 0 && s.charAt(i) == 'U') valleys++;
    }
    return valleys;
}
```

```
static String represent(boolean r) {
  return r ? "YES" : " NO";
}

private static boolean solve(int x1, int v1, int x2, int v2) {
  double v = (double) (x2-x1) / (v1-v2);
  return (int) v == v;
}

private static boolean canBeSolved(int x1, int v1, int x2, int v2) {
  return (x2>x1 && v1>v2) || (x2<x1 && v1<v2) || (x1==x2 && v1==v2);
}

static String kangaroo(int x1, int v1, int x2, int v2) {
  boolean r = canBeSolved(x1, v1, x2, v2) && solve(x1, v1, x2, v2);
  return represent(r);
}
```

$s = v \cdot t$

1. I want to brake down complexity
2. - split implement and result represent
3. I want to get rid of values which can't be met

we enriched our code with extra knowledge

solution split between different fn's

Page 9

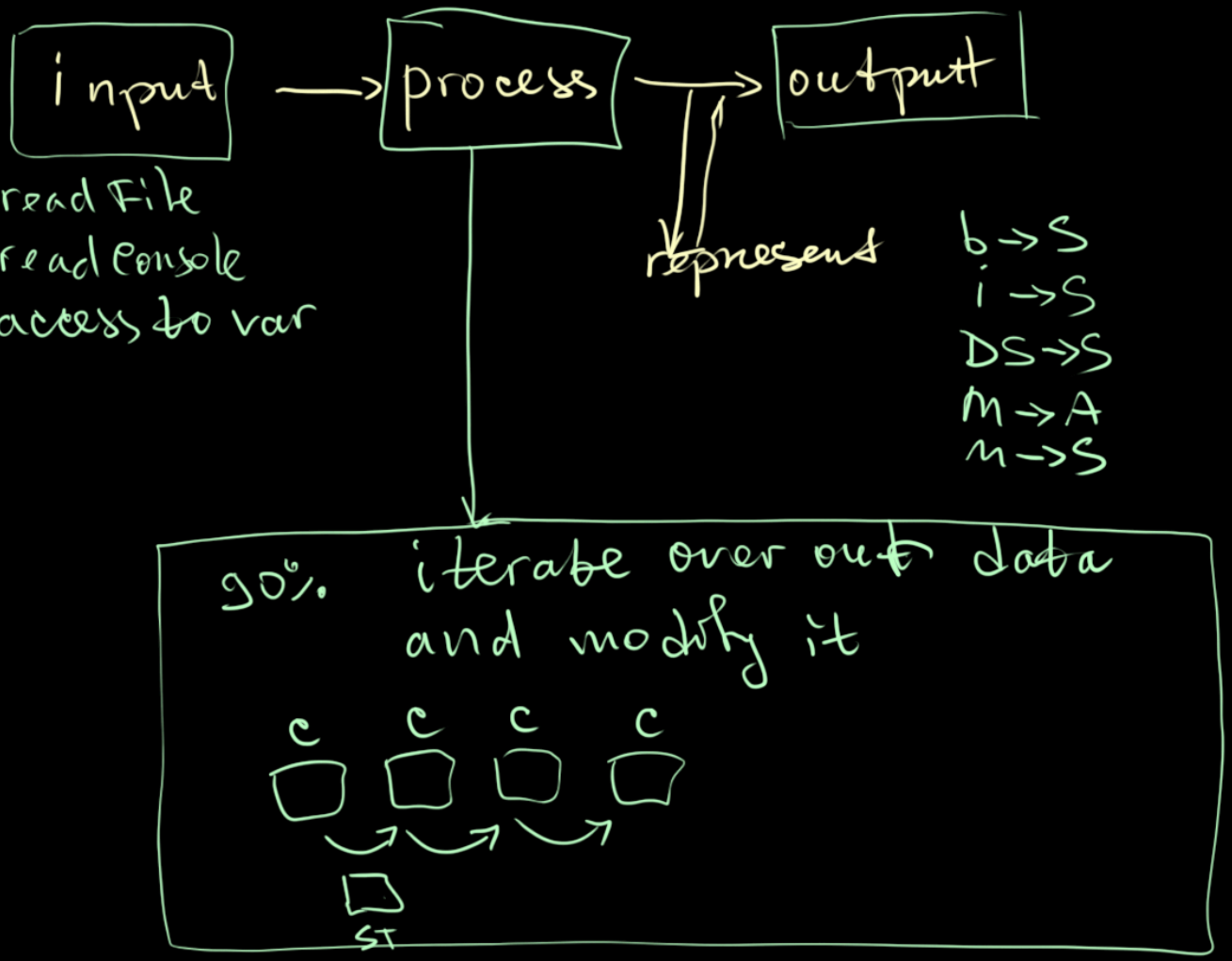
```
static class State {
  int level = 0;
  int prev = 0;
  int count = 0;
}

static int countingValleys(int n, String s) {
  State st = new State();

  IntConsumer process = c -> {
    switch (c) {
      case 'D': st.prev=st.level; st.level--; break;
      case 'U': st.prev=st.level; st.level++; break;
      default : throw new RuntimeException("should be here ;)");
    }
    if (st.level==0 && st.prev<0) st.count++;
  };

  s.chars().forEach(process);
  return st.count;
}
```

we have all variables encapsulated in one place

prefer to use CASE over IF

we separated concerns

that's only line where we know, that we deal w/ string



I don't want to spread my variables through whole code

Relations, Constraints, Foreign Keys

1:1
1:N
M:N

student

1:1
1:1

name
age
=
=
=

mandatory fields columns

email
phone
extra

optional

10K

SELECT * FROM STUDENTS
LEFT OUTER JOIN STEXTRA
on (s.id = sx.id)

data and relation are fused

students

id	age	name	sx_id
1		Jim	<null>
2		Jack	
3		Alex	1
4		Ben	

students_extra

id	extra
1	extra 10K.....

- we have a lot of nulls
- we waste the memory
- we need to modify stud. table

FOR SX. we always provide manually (from students)

students L

id	age	name
1		Jim
2		Jack
3		Alex
21		Ben

1:1

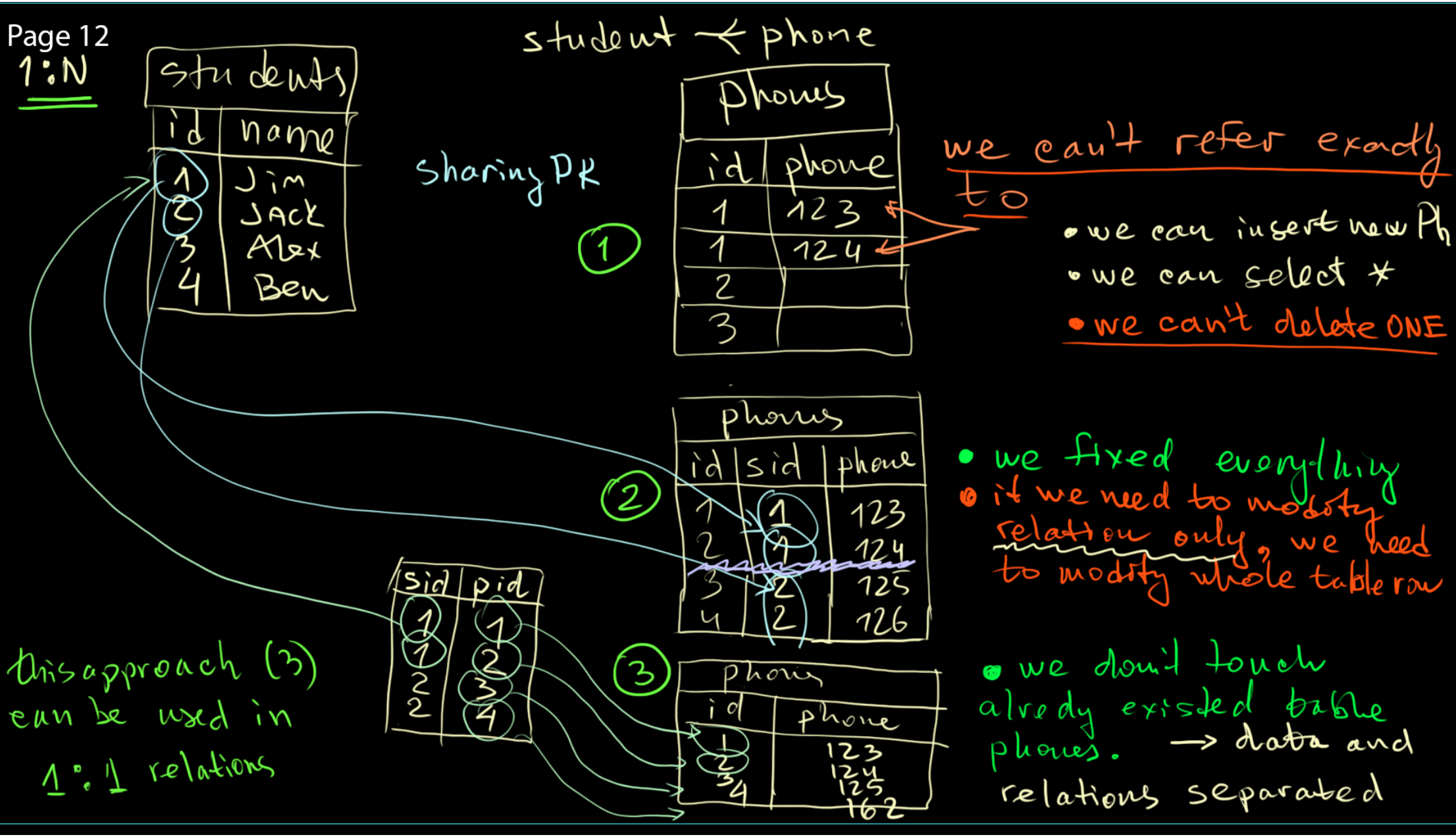
- we don't have nulls
- shared primary key
- because we don't need extra key

student_extra R

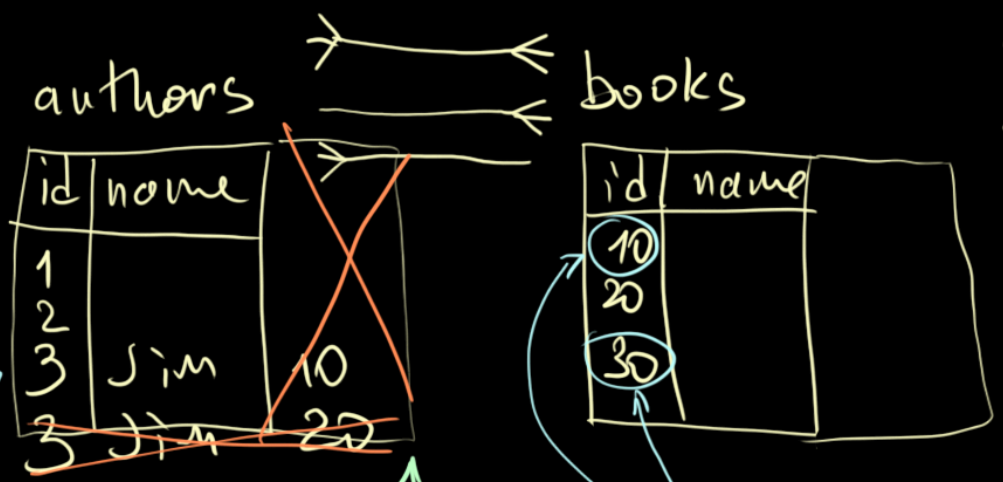
id	extra
2	Jack X
4	Ben X

PK

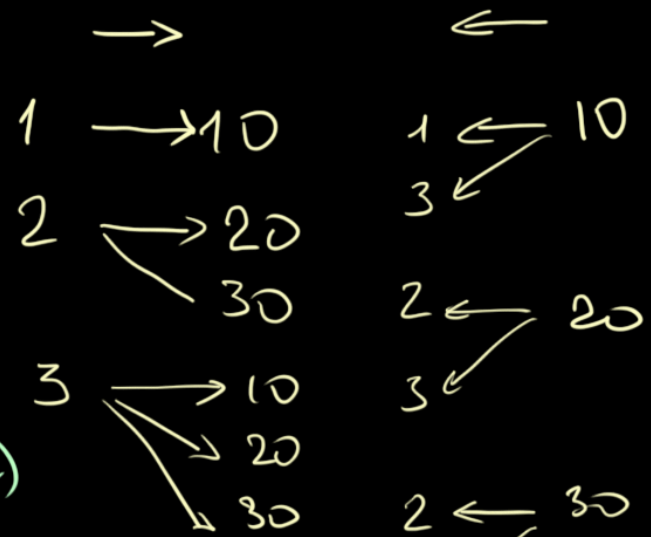
1:N


student < phone
Students
id | name
1 | Jim
2 | JACK
3 | Alex
4 | Ben
Sharing PK
1
Phones
id | phone
1 | 123
1 | 124
2
3
we can't refer exactly to
• we can insert new Ph
• we can select *
• we can't delete ONE
2
phones
id | sid | phone
1 | 1 | 123
2 | 1 | 124
3 | 2 | 125
4 | 2 | 126
• we fixed everything
• if we need to modify relation only, we need to modify whole table row
sid | pid
1 | 1
1 | 2
2 | 3
2 | 4
3
phones
id | phone
1 | 123
2 | 124
3 | 125
4 | 162
this approach (3) can be used in 1:1 relations
• we don't touch already existed table phones. → data and relations separated

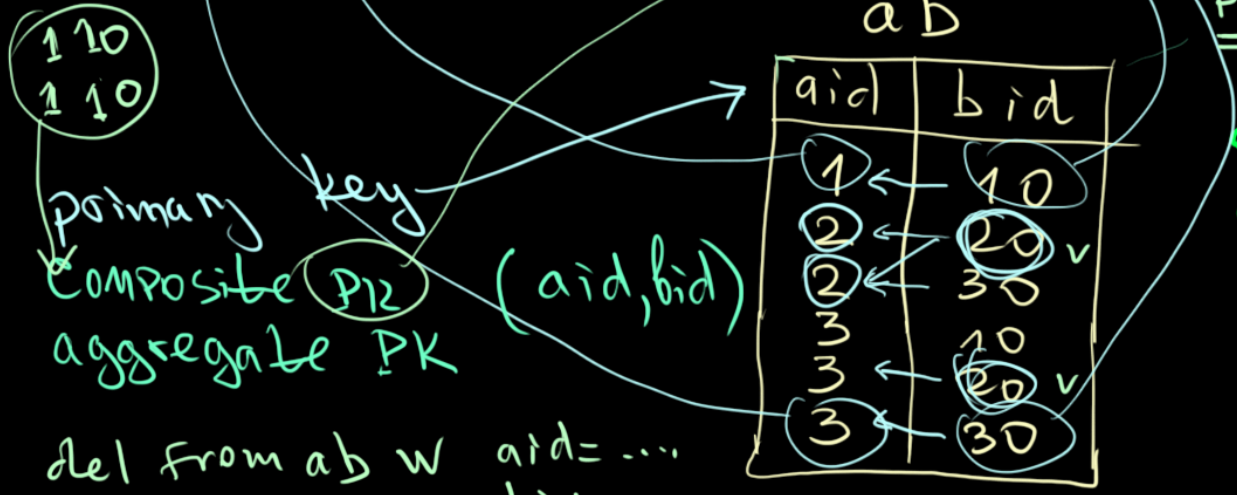
M:N



- one Author has many books
- one Book has MANY Authors



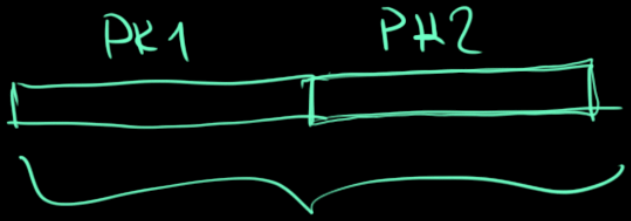
we should think that PK = f(PK1, PK2) unique



- we separated data and 3 relations
- we can easily address (refer) to any relation
- everything is flexible
- we need extra JOIN in SQL

PK1 id1 0..256 → 1 byte
PK2 id2 0..256 → 1 byte

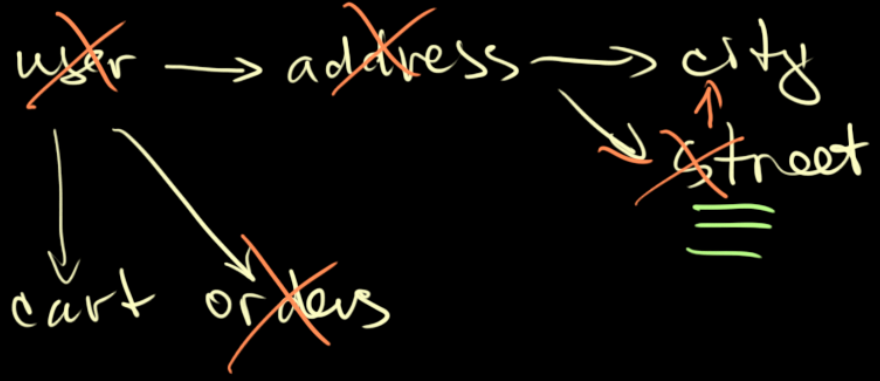
aggregated primary key



PK = PK1 << 8 + PK2

SizeOf(PK) = SizeOf(PK1) + SizeOf(PK2)

- if we want to delete the record from T1
  - deny, because it belongs to another relation
  - you can delete all linked relations (!!! be extremly careful)
  - set null on other relations



- to compare streets1 and streets2
- to add new streets to the table

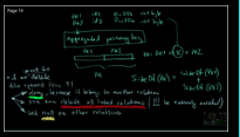
```
DELETE * FROM STREETS
INSERT * INTO STREETS
```

So, everything was delete

Q: Why deleted streets caused deleted orders, bcause we deleted data only

A: because DB was set up with wrong FK strategy

↳ delete all connected relations



	id	name
1	1	Jim
2	2	Jack
3	3	Alex
4	4	Ben

	id	extra
1	2	JackExtra
2	4	BenExtra

1 : 1

```
SELECT * from "user" u
join uextra x on (u.id=x.id)
```

2 rows

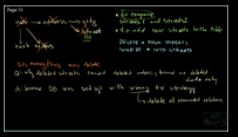
	u.id	name	x.id	extra
1	2	Jack	2	JackExtra
2	4	Ben	4	BenExtra

```
SELECT * from "user" u
left outer join uextra x on (u.id=x.id)
```

	u.id	name	x.id	extra
1	1	Jim	<null>	<null>
2	2	Jack	2	JackExtra
3	3	Alex	<null>	<null>
4	4	Ben	4	BenExtra

u.id: int

we don't have ——— in our data.
we have them only in results



1:N

	id	name
1	1	Jim
2	2	Jack
3	3	Alex
4	4	Ben

	id	phone	uid
1	1	123	1
2	2	124	1
3	3	125	2

```
select * from "user" u
join phones2 p on (p.uid=u.id)
```

	u.id	name	p.id	phone	uid
1	1	Jim	1	123	1
2	1	Jim	2	124	1
3	2	Jack	3	125	2

I want to see the students who has a phone

```
select * from "user" u
left outer join phones2 p on (p.uid=u.id)
```

	u.id	name	p.id	phone	uid
1	1	Jim	1	123	1
2	1	Jim	2	124	1
3	2	Jack	3	125	2
4	4	Ben	<null>	<null>	<null>
5	3	Alex	<null>	<null>	<null>

I want to see the ALL students with their phones

if you have chances that part of the data is missed on joining table

you need to use left outer join

```
select distinct u.id, u.name from "user" u
join phones2 p on (p.uid=u.id)
```

	id	name
1	1	Jim
2	2	Jack

MiN

id	name
1	Jim
2	Alex
3	Ben

id	name
10	Java
20	Scala
30	Haskell

aid	bid
1	10
2	20
2	30
3	10
3	20
3	30

ab
- aid integer
- bid integer
- ab_pk (aid, bid)
- ab___fk1 (aid) → author (id)
- ab___fk2 (bid) → book (id)
- ab_pk (aid, bid) UNIQUE

composite PK

L
R
R

R
L
R

R
R
L

Author → Book

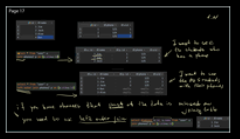
Book → Author

```
select * from author a
left outer join ab on (ab.aid=a.id)
left outer join book b on (b.id=ab.bid)
```

a.id	a.name	aid	bid	b.id	b.name
1	Jim	1	10	10	Java
2	Alex	2	20	20	Scala
2	Alex	2	30	30	Haskell
3	Ben	3	10	10	Java
3	Ben	3	20	20	Scala
3	Ben	3	30	30	Haskell

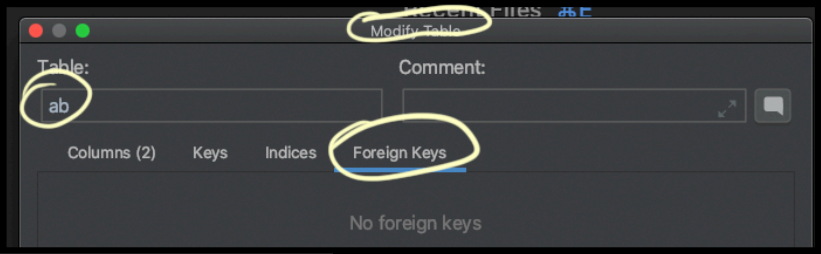
```
select a.id, a.name, b.id, b.name from author a
left outer join ab on (ab.aid=a.id)
left outer join book b on (b.id=ab.bid)
```

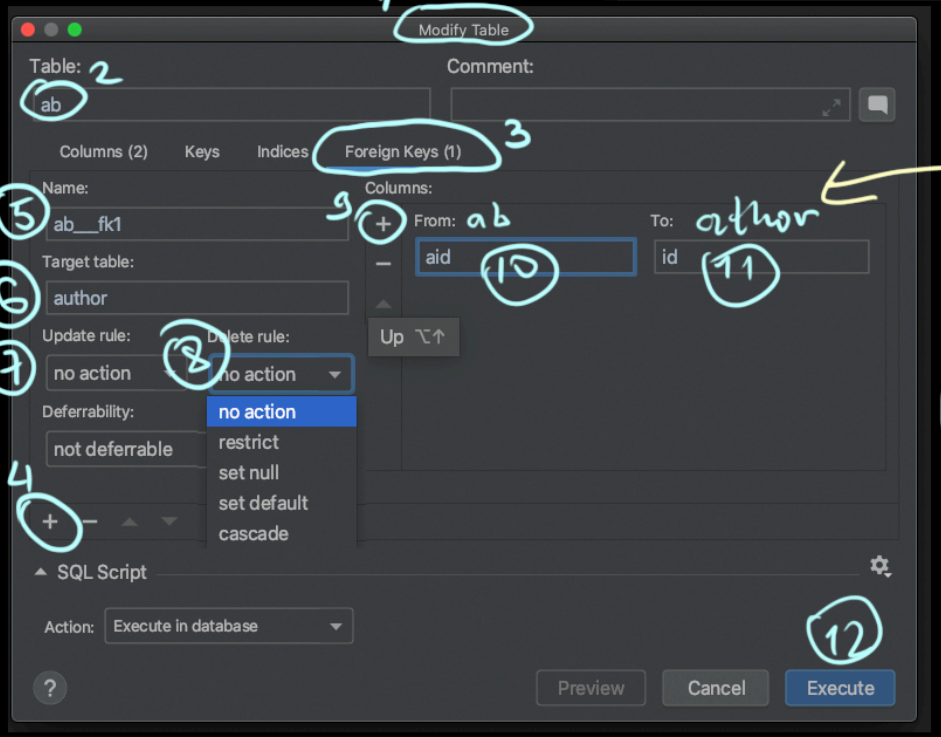
a.id	a.name	b.id	b.name
1	Jim	10	Java
2	Alex	20	Scala
2	Alex	30	Haskell
3	Ben	10	Java
3	Ben	20	Scala
3	Ben	30	Haskell

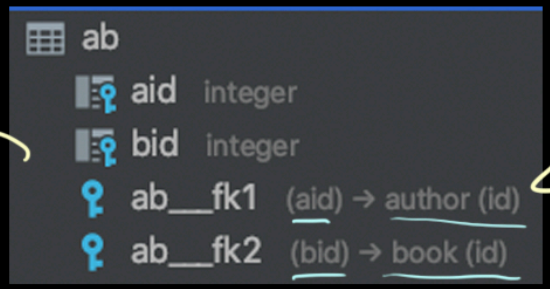


Page 19

Foreign Keys








```
select * from ab
join author a on ab.aid = a.id
join book b on ab.bid = b.id
```

is offered automatically because of

you write

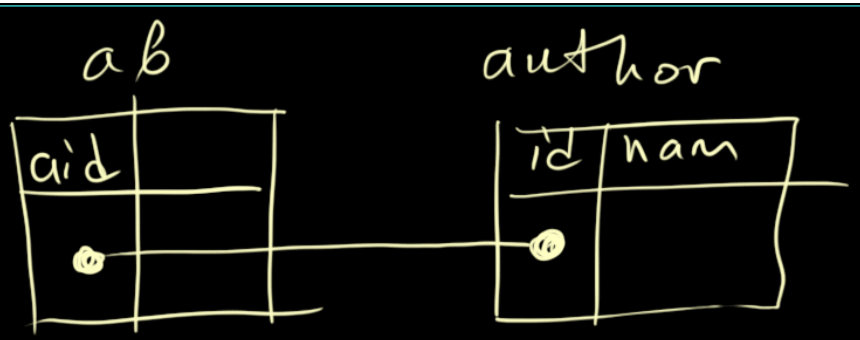
```
sql-lesson.public> DELETE FROM public.author WHERE id = 2
[2020-04-29 11:12:40] [23503] ERROR: update or delete on table "author" violates foreign key constraint "ab___fk1" on table "ab"
[2020-04-29 11:12:40] Detail: Key (id)=(2) is still referenced from table "ab".
```

```
1 alter table ab
2     add constraint ab___fk1
          foreign key (aid) references author
```

just a name

```
1 alter table ab
2     add constraint ab___fk2
          foreign key (bid) references book
```

type of constraint



after that constraint you can't delete author, book if they are exist in ab table

to keep your data consistent.

- alter table ab fk(aid) ref author (id)
- alter table author fk (id) ref ab (aid)