Examples of DIVISION – RELATIONAL ALGEBRA and SQL

r ÷ s is used when we wish to express queries with "all":

Ex. "Which persons have a loyal customer's card at ALL the clothing boutiques in town X?"
"Which persons have a bank account at ALL the banks in the country?"
"Which students are registered on ALL the courses given by Soini?"
"Which students are registered on ALL the courses that are taught in period 1?"
"Which boys are registered on those courses that are taken by ALL the girls?"
"Which girls are registered on ALL the courses taken by student nr. 40101?"

In all these queries the description after ALL defines a set with a number of elements. The result is composed of those data units (persons, students etc.) who satisfy these requirements. The logic these sentences express is that of **implication**: for which persons is it true that

"**IF** there is a clothing boutique in town X **THEN** the person has a loyal customer's card there",
"**IF** Soini gives a course **THEN** the student is registered on it" or
"**IF** 40101 is registered on a course **THEN** the girl is registered on the same course".

We shall study the last example in detail here. On the one hand we have a list (a relation) with girls and the courses they are registered on: this will be called the relation r. On the other hand we have a list (a relation) of ALL the courses taken by 40101. This will be the relation s. Now we wish to know which of the girls take ALL these courses. Suppose that 40101 takes the following three courses: G555, 456306 and 456302. To be a part of the answer the girl in question must be registered on ALL of these three. (In addition to that she may be registered on other courses not taken by 40101, these will not affect the result.)

In relational algebra the query will be expressed as follows (the example database "kursdatabas" is found on the home page of this course. *matr* = student number, *namn* = name, *kurskod* = course code, *kön* = sex, *kursanmälan* = course registration):

$$\Pi_{\text{matr, namn, kurskod}} (\sigma_{\text{kön = 'K'}} \text{(student) |x| kursanmälan}) \div \Pi_{\text{kurskod}} (\sigma_{\text{matr = 40101}} \text{(kursanmälan)})$$

The result will be a relation with the attributes *namn* and *matr*. The attribute *kurskod* that we are dividing by will "disappear" in the division.

NOTE! "Which persons have passed ALL the courses they have registered on?" has the same surface form, but a different logic: the set of courses that is tested will vary from one person to another, depending on which courses the current person has registered on. There is **no common set** of courses that should be tested for each person. The formulation in SQL will often be easier than that using division; to express the example query it is enough to create the set of persons registered on courses and then subtract those who have NULL as the value of some course mark (*vitsord*, one of the attributes in *kursanmalan* in the sql-code of the example database).

Which female students take **ALL** the courses that 40101 is taking?

$\Pi_{\text{matr, namn, kurskod}} (\sigma_{\text{kön = 'K'}} (\text{student}) \; |x| \; \text{kursanmälan})$ $\quad \Pi_{\text{kurskod}} (\sigma_{\text{matr = 40101}} (\text{kursanmälan}))$

matr	namn	up	kön	kurskod
40112	Brita	DT	K	456306
40112	Brita	DT	K	456302
40113	Ann-Helen	DT	K	456304
40113	Ann-Helen	DT	K	456306
40113	Ann-Helen	DT	K	456302
40128	Siru	DT	K	G555
40128	Siru	DT	K	456306
40128	Siru	DT	K	456302
40240	Sara	IS	K	456304
40240	Sara	IS	K	456306
40240	Sara	IS	K	456302

matr	kurskod
40101	G555
40101	456306
40101	456302

$\Pi_{\text{matr, namn, kurskod}} (\sigma_{\text{kön = 'K'}} (\text{student}) \; |x| \; \text{kursanmälan}) \div \Pi_{\text{kurskod}} (\sigma_{\text{matr = 40101}} (\text{kursanmälan}))$

matr	namn	up	kön	kurskod
40112	Brita	DT	K	456306
40112	Brita	DT	K	456302
40113	Ann-Helen	DT	K	456304
40113	Ann-Helen	DT	K	456306
40113	Ann-Helen	DT	K	456302
40128	Siru	DT	K	G555
40128	Siru	DT	K	456306
40128	Siru	DT	K	456302
40240	Sara	IS	K	456304
40240	Sara	IS	K	456306
40240	Sara	IS	K	456302

matr	kurskod
40101	G555
40101	456306
40101	456302

We can see that only 40128, Siru has all the required courses.

$r \div s$ is defined as $\Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

$r \longleftarrow \Pi_{\text{matr, namn, kurskod}} (\sigma_{\text{kön = 'K'}} (\text{student}) \;|x|\; \text{kursanmälan})$

$s \longleftarrow \Pi_{\text{kurskod}} (\sigma_{\text{matr = 40101}} (\text{kursanmälan}))$

$\Pi_{R-S}(r)$ is here $\Pi_{\text{matr, namn}}(r)$ (No duplicate elements in sets!)

$\Pi_{R-S,S}(r)$ is here $\Pi_{\text{matr, namn, kurskod}}(r)$, in this case = r. The projection will guarantee that the attributes come in the right order for the subtraction.

$\Pi_{R-S}(r) \times s$ is here $\Pi_{\text{matr, namn}}(r)$ x {G555, 456306, 456302}, i. e. all the possible combinations of a girl registered on some course and the courses taken by 40101. From this set we shall subtract those tuples that represent real elements (actual course registrations). From what is left (the combinations representing “false”, unactualized information, we shall project the student number (*matr*) and the name (*namn*), the attributes represented by (Π_{R-S}). Finally these will be subtracted from $\Pi_{\text{matr, namn}}(r)$. The answer is what is left.

$\Pi_{R-S}(r)$

==============

~~40112~~	~~Brita~~
~~40113~~	~~Ann-Helen~~
40128	Siru
~~40240~~	~~Sara~~

SVAR:

=====

40128 Siru

The “false” candidates have been stricken through.

$\Pi_{R-S}(r) \times s$ (tänkbara)

=====================

40112	Brita	G555
~~40112~~	~~Brita~~	~~456306~~
~~40112~~	~~Brita~~	~~456302~~
40113	Ann-Helen	G555
~~40113~~	~~Ann-Helen~~	~~456306~~
~~40113~~	~~Ann-Helen~~	~~456302~~
~~40128~~	~~Siru~~	~~G555~~
~~40128~~	~~Siru~~	~~456306~~
~~40128~~	~~Siru~~	~~456302~~
40240	Sara	G555
~~40240~~	~~Sara~~	~~456306~~
~~40240~~	~~Sara~~	~~456302~~

The tuples representing true information have been stricken through.

$\Pi_{R-S,S}(r)$ (verkliga)

=====================

40112	Brita	456306
40112	Brita	456302
40113	Ann-Helen	456304
40113	Ann-Helen	456306
40113	Ann-Helen	456302
40128	Siru	G555
40128	Siru	456306
40128	Siru	456302
40240	Sara	456304
40240	Sara	456306
40240	Sara	456302

(Tuples that are not marked with green represent real information about other courses than the relevant ones.)

How to translate this in MySQL?

Problem: There is neither division (÷) nor set difference (-) i MySQL, so you are obliged to find other ways around this problem. There are (at least) three ways to express these queries:

1) Not exists (¬∃) combined with not in (∉): ("There may **not** be a course that 40101 takes that is **not** among the courses taken by the girl in question")

```
select distinct matr, namn
from student as R
where kon = 'K' and
      not exists (select kurskod
                  from kursanmalan
                  where matr = 40101 and
                        kurskod not in (select kurskod
                                        from kursanmalan as R2
                                        where R.matr = R2.matr));
```

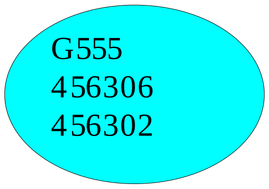
```
+--------+------+
| matr   | namn |
+--------+------+
| 40128  | Siru |
+--------+------+
1 row in set (0.00 sec)
```

The outmost selection will decide which columns (matr, namn) we shall see in the result. The outmost `select – from – where` will create a table including all the female students and their course registrations. These students will now be tested one by one: for the female student to be part of the result, the not exists-clause for this student must be true (the list after the not exists must remain empty).

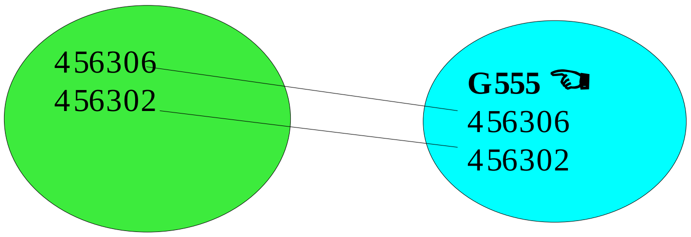
What does this list consist of? First we put there all the courses that the student 40101 is registered on (3 items: G555, 456306 and 456302). These are represented by kurskod. Then these courses are tested one by one against the courses the current female student is registered on: a set of her courses is created in the innermost `select`. These are represented by kurskod. This set is thus composed of **all** the courses that the female student is registered on. As soon as we find a kurskod among 40101's courses that is not among the course codes of the current girl, this kurskod will part of the `select` list in the middle, which accordingly will no longer be empty. This in turn leads to the not exists becoming false, and that means that the female student in question does not satisfy the requirements (she does not have all the courses required) and she will not be part of the result. On the other hand, if all the course codes in the `select` in the middle are found in the set of the course codes taken by the current female student, this `select` will stay empty and the not exists will accordingly be true. Now this student will be part of the result.

The example query:

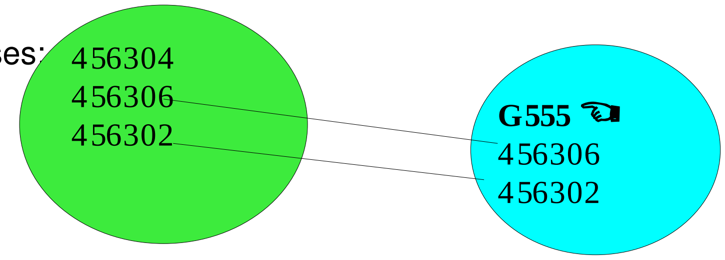
40101's courses:



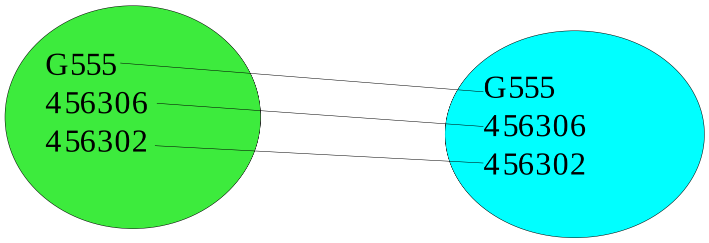
Brita's (40112) courses:



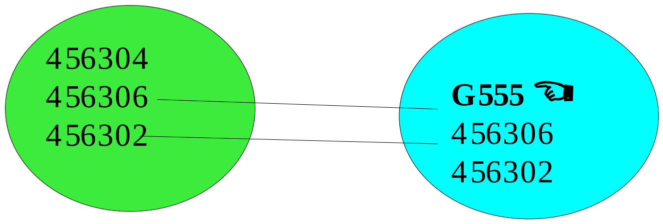
Ann-Helen's (40113) courses:



Siru's (40128) courses:



Sara's (40240) courses:



☜ Shows which course(s) are **lacking** in the girls' selections. These courses will be part of the selection in the middle and make not exists false. If **all** the courses taken by 40101 can be found among the courses chosen by the girl, she will be part of the result.

2) Two times not exists: (“There may **not** be a course that 40101 takes that is **not** taken by the current girl”)

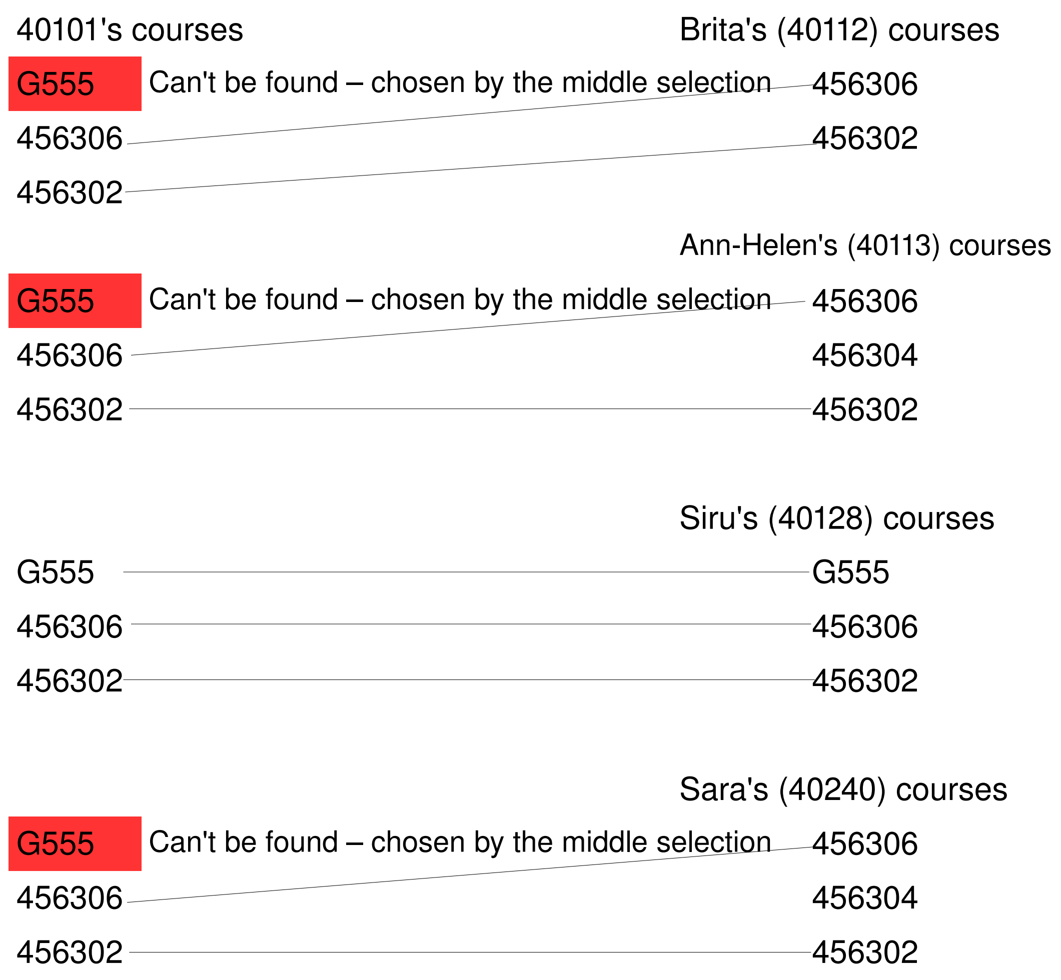
```
select distinct matr, namn
from student as R
where kon = 'K' and
      not exists (select kurskod
                  from kursanmalan as S
                  where matr = 40101 and
                        not exists (select kurskod
                                    from kursanmalan as R2
                                    where R.matr = R2.matr and
                                          S.kurskod = R2.kurskod ));
```

```
+--------+------+
| matr   | namn |
+--------+------+
| 40128  | Siru |
+--------+------+
1 row in set (0.01 sec)
```

This solution is very similar to the last one, but now the test with set membership is replaced by another not exists. The outer not exists will choose the courses taken by 40101. The inner not exists will control these courses one by one; will we find the same course for the girl in question? If there is a kurskod (among the courses taken by the girl) equal to the kurskod (of 40101's courses) that we are testing at the moment, this kurskod will be the result of the innermost select. In this case this selection will not be empty, and the inner not exists will thus be false. Now the kurskod we just tested will **not** be a part of the selection in the middle.

Each kurskod for 40101 will be tested like this. If we can find them **all** among the courses taken by the girl, none of them will be selected in the middle and the outer not exists will thus be true. This means that the girl in question had all the specified courses and that she will accordingly be part of the result. On the other hand, if there are one or several course codes that will not find their counterpart in the innermost selection, this selection will be empty for them, making the inner not exists true. These course codes will be a part of the select in the middle, making the outer not exists false, the girl in question (who did not have these courses) will **not** be a part of the result.

See the figure on next page!


40101's courses
Brita's (40112) courses
G555
Can't be found – chosen by the middle selection
456306
456302
456306
456302
Ann-Helen's (40113) courses
G555
Can't be found – chosen by the middle selection
456306
456302
456306
456304
456302
Siru's (40128) courses
G555
456306
456302
G555
456306
456302
Sara's (40240) courses
G555
Can't be found – chosen by the middle selection
456306
456302
456306
456304
456302

Next page!

3) “Count and compare”

The idea behind this method is to count how many different values there are in the divisor (i. e. all the values that one should have to be a part of the result). When this is applied to our example we first count how many courses 40101 takes. Then we count how many of **these** courses our female students take. If we get the same result, the girl in question has all the courses necessary and will appear in the result.

```
mysql> select matr, namn
       from kursanmalan natural join student
       where kon = 'K' and
             kurskod in (select kurskod
                         from kursanmalan
                         where matr = 40101)
       group by matr
       having count(kurskod) = (select count(*)
                                from (select kurskod
                                      from kursanmalan
                                      where matr = 40101) as tab);
```

```
+-------+------+
| matr  | namn |
+-------+------+
| 40128 | Siru |
+-------+------+
1 row in set (0.00 sec)
```

Here is the result of **`kursanmalan natural join student`** `where kon = 'K' and kurskod in (select kurskod from kursanmalan where matr = 40101):`

```
+-------+---------+---------+-----------+------+------+
| matr  | kurskod | vitsord | namn      | up   | kon  |
+-------+---------+---------+-----------+------+------+
| 40112 | 456302  |    NULL | Brita     | DT   | K    |
| 40112 | 456306  |    NULL | Brita     | DT   | K    |
| 40113 | 456302  |    NULL | Ann-Helen | DT   | K    |
| 40113 | 456306  |    NULL | Ann-Helen | DT   | K    |
| 40128 | 456302  |    NULL | Siru      | DT   | K    |
| 40128 | 456306  |    NULL | Siru      | DT   | K    |
| 40128 | G555    |    NULL | Siru      | DT   | K    |
| 40240 | 456302  |    NULL | Sara      | IS   | K    |
| 40240 | 456306  |    NULL | Sara      | IS   | K    |
+-------+---------+---------+-----------+------+------+
9 rows in set (0.00 sec)
```

Next page!

These tuples are now grouped according to the student number:

```
+-------+---------+---------+-----------+------+------+
| matr  | kurskod | vitsord | namn      | up   | kon  |
+-------+---------+---------+-----------+------+------+
| 40112 | 456302  |    NULL | Brita     | DT   | K    |
| 40112 | 456306  |    NULL | Brita     | DT   | K    |
| 40113 | 456302  |    NULL | Ann-Helen | DT   | K    |
| 40113 | 456306  |    NULL | Ann-Helen | DT   | K    |
| 40128 | 456302  |    NULL | Siru      | DT   | K    |
| 40128 | 456306  |    NULL | Siru      | DT   | K    |
| 40128 | G555    |    NULL | Siru      | DT   | K    |
| 40240 | 456302  |    NULL | Sara      | IS   | K    |
| 40240 | 456306  |    NULL | Sara      | IS   | K    |
+-------+---------+---------+-----------+------+------+
```

`having count(kurskod)` counts the number of course codes/group, i. e. the number of courses each of these girls has registered on (note that in this selection we are only concerned with those courses that 40101 takes; it is quite possible that the girls are also registered on other courses in addition to these).

```
having count(kurskod) = (select count(*)
                         from (select kurskod
                               from kursanmalan
                              where matr = 40101) as tab);
```

compares the number of courses/group (the number of the relevant courses taken by the girl in question) with the number of courses taken by 40101 (3 courses). If we get the **same number**, we know that we have the **same courses**, and then the girl will appear in the result. This happens in the selection in the beginning of the query:

```
mysql> select matr, namn
       from kursanmalan natural join student
       where kon = 'K' and
             kurskod in (select kurskod
                         from kursanmalan
                         where matr = 40101)
       group by matr
       having count(kurskod) = (select count(*)
                                from (select kurskod
                                      from kursanmalan
                                      where matr = 40101) as tab);
```

OBS! Here we can only have one registration/course. In other cases we might need to use **count distinct to e**liminate duplicates.

OBS! It would be a natural solution to use a temporary table to represent the courses taken by

40101. Unfortunately, the same temporary table may not be opened twice in MySQL 5.1, so that solution will only give you error messages.