Problema 4. Re-Proiectare BD - 3 săptămâni

Sunt situații când în urma re-analizării specificațiilor inițiale sau ca urmare a unor modificări de cerințe, să se constate că trebuie modificate multiplicitățile relațiilor dintre anumite tabele. Prin urmare structura bazei de date trebuie revizuită.

Pornind de la baza de date definită în cadrul *Temei 1*, identificați în baza de date 3 relații distincte de tip 1:N și o relație de tip M:N. Tabelele implicate în aceste relații trebuie să fie distincte. Introduceți **minimum 10 înregistrări** în fiecare tabel, astfel încât relațiile să fie efectiv reprezentate în date.

Efectuați următoarele transformări asupra relațiilor identificate:

- transformați o relație **1:N** în **N:1**;
- transformați o relație **1:N** în **M:N**;
- transformați o relație **M:N** în **1:N**;
- transformați o relație **1:N** în **1:1**.

Reguli de păstrare a datelor

- în cazul în care, în urma modificării, o înregistrare face referire la mai multe alte înregistrări, se va păstra cheia externă corespunzătoare înregistrării cu cea mai mare valoare.
- Valorile cheilor primare ale înregistrărilor pentru care a fost necesară ștergerea legăturilor, se vor salva într-un **tabel nou creat** avand urmatoarele campuri: NumeTabelSt, IdSt, NumeTabelDr, IdDr, unde primele doua campuri reprezinta numele si cheia primara a tabelei din stanga relatiei, iar ultimele doua numele si cheia primara a tabelei din dreapta relatiei (de exemplu, *Legaturi_Eliminate*).

Observație:
Dacă în schema proiectată (din Tema 1 + modificările din Tema 3) nu există toate relațiile cerute între tabele distincte (cele 3 relații 1:N și relația M:N), veți adăuga tabele suplimentare și/sau coloane de legătură astfel încât să obțineți relații cu multiplicitățile respective, menținând tabelele implicate distincte.

Indications:

Maximum grade is **10** for a **generic solution**.

- Write one or multiple stored procedure(s) to change the multiplicity between tables A and B, given as generic parameters; use system tables / functions to search the required information and perform the modification, for any two tables.

Maximum grade is **9** for a **hardcoded solution**.

- Write one or multiple stored procedure(s) to change the multiplicity between particular tables A and B, in which are effectively used the **names of the tables** (such as, *Ingredients*, *Teas*), and the **type of each attribute**, **name** of **primary key** (PK…) **constraint**, **name** of

foreign key (FK…) **constraint**. – the stored procedure can have as many parameters as desired.

Maximum grade is **9.25 – 9.5 - 9.75** (0.25p per each multiplication) for a **hybrid solution** (partial generic and partial hardcoded).

It is a task what requires both DDL (Data Definition Language) and DML (Data Manipulation Language).

Please make *back-up's* of the database. For example, before start, after the first multiplicity change, after the second multiplicity change, after the third multiplicity change, after the last multiplicity change.

Use ***Restore database*** to be able to return to what you had previously in the database.

Steps to follow:

Pornind de la baza de date definită în cadrul *Temei 1*, identificați în baza de date 3 relații distincte de tip 1:N și o relație de tip M:N. Tabelele implicate în aceste relații trebuie să fie distincte. Introduceți **minimum 10 înregistrări** în fiecare tabel, astfel încât relațiile să fie efectiv reprezentate în date.

A. Make sure you have 3 distinct 1-N relationships and one M-N relationship.

- If need it, create extra tables connected directly with your database.
- e.g. *Address* – Person (1-N) – Address is the new added table and it's primary key is added in Person as foreign key (the same type and values).
- e.g. *TypeOfBook* – Book (1-N) – TypeOfBook is the new added table and it's primary key is added in Book as foreign key (the same type and values).
- e.g. Person – *Review* (1-N) – Review is the new added table and has a column that is foreign key referencing primary key from Person.

Distinct relationships: unique tables
Each multiplicity change is unique and does not reflect in the others multiplicities.
For example: see the database diagram below

- *Ingredients* – *Teas* (1-N)
- *Posibilities* – *Teas* (1-N) - **NOT GOOD** because it is using the same "Teas" table as "N" side of the relationship; it is consider here to observe that the multiplicity change "1-N to N-1" does not effect the multiplicity change "1-N to M-N" or any other.
- *Managers* – *Shops* (1-N)
- *Shops* – *Teas* (M-N)

The relationships considered can be in the same side of the diagramo or in various sides.

B. Create an .sql file in which is performed the ***insert*** in the relations 1-n, and m-n considered.

Mandatory, because this is how the database is used in real-life, with records and the coresponding constraints. At least **10 records** in each table considered.

Possible solutions:

- Simple ***Insert***'s in primary key table first and then in corresponding foreign key table.
- Insert into ... Go 10 (if the PK is identity) – 10 identical recors will be added
- Stored procedures for *Insert* in PK table and FK table
- ... any solution

Reguli de păstrare a datelor

- în cazul în care, în urma modificării, o înregistrare face referire la mai multe alte înregistrări, se va păstra cheia externă corespunzătoare înregistrării cu cea mai mare valoare.
    - Refers to DML – update, insert (especially) – when a relationship is changed related to multiplicity, some values are lost, and even if the relationship is recreated there are still no records (as FK)... so, practilly the data (values) should be also restored. Is recommended to keep the maximum value because can offers a feedback related to the number of records that are in the new relationship.
    - For example, *Posibilities – Teas* (1-n) with the following records

Posibilities

Sid	***Type***	***NoOfTeas***
1	black	10
2	green	8
3	plants	12

Teas

Cid	***Name***	***Quantity***	***Price***	***Sid***	***Iid***
1	B1	10	12	1	1
2	B2	5	15	1	2
3	B3	6	14	1	1
4	G1	9	16	2	3
5	G2	1	17	2	4
6	P1	2	8	3	4
7	P2	5	7	3	5

Due to this rule, in this case, we will keep the foreign key corresponding to the maximum value.

3 – for the FK *Sid=**1***

3	B3	6	14	1	1

5 – for the FK *Sid=**2***

5	G2	1	17	2	4

7 – for the FK *Sid=**3***

7	P2	5	7	3	5

and the rest of the values will get lost (deleted).

- Valorile cheilor primare ale înregistrărilor pentru care a fost necesară ștergerea legăturilor, se vor salva într-un **tabel nou creat** avand urmatoarele campuri: NumeTabelSt, IdSt, NumeTabelDr, IdDr, unde primele doua campuri reprezinta numele si cheia primara a tabelei din stanga relatiei, iar ultimele doua numele si cheia primara a tabelei din dreapta relatiei (de exemplu, *Legaturi_Eliminate*).

C. Create table **New_table(NameTableLeft, IdLeft, NameTableRight, IdRight)**

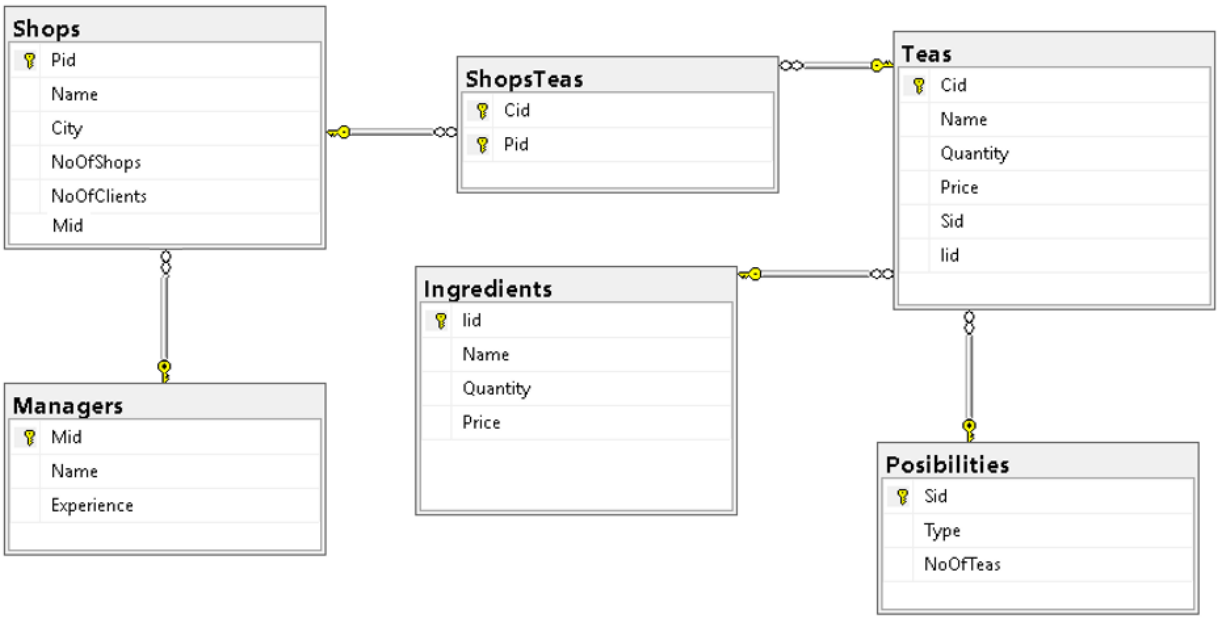
- **NameTableLeft** – **table name** from the **left** side of the relation considered
- IdLeft – primary key of the left side table
- **NameTableRight** - table name from the **right** side of the relation considered
- IdRight - primary key of the right side table

This table has to been seen as a **log table** that keeps the history of all operations performed. Maybe, a better structure can be:

RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription*, MultiplicityDate)

e.g. *MultiplicityDescription** attribute can contain a suggestive phrase for the multiplicity solved, such as *relationship "1-N Ingredients – Teas has been changed to relationship N-1 Ingredients – Teas"*, or more simple (if it is possible to have a generic writting).
e.g. *MultiplicityDate* attribute will store the actual date of execution, such as *getdate(), current_date() – sysdatetime(), ...*

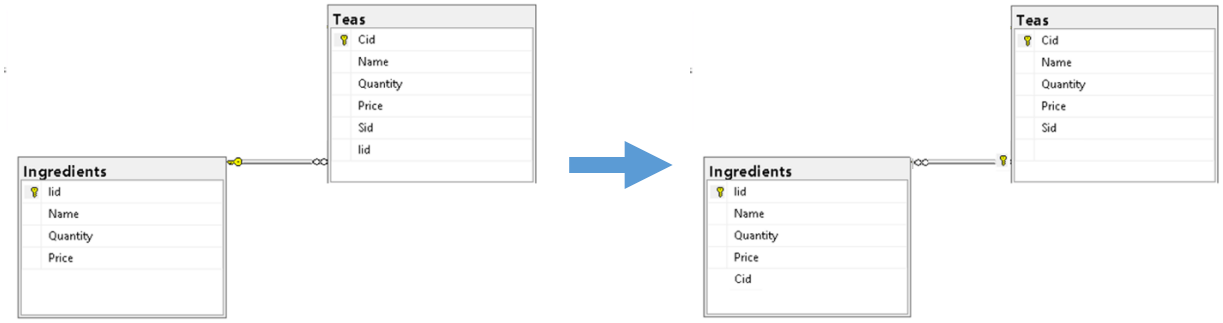
For a more concrete discussion, let us consider the following database



One need to work with **ALTER TABLE** statements.

1. transformați o relație **1:N** în **N:1**;

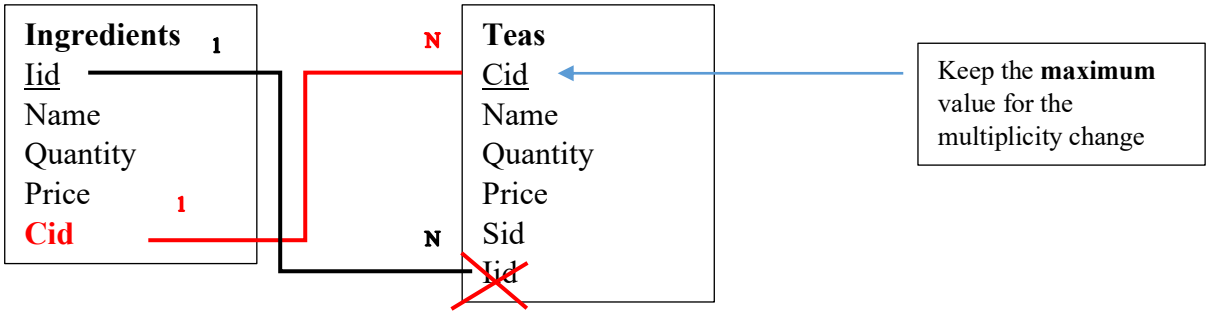
1:N – **Ingredients – Teas** (an ingredient is used in multiple teas and a tea is using a single ingredient) will be transformed in **N:1 – Ingredients – Teas** (an ingredient is used in a single tea and a tea contains multiple ingredients)

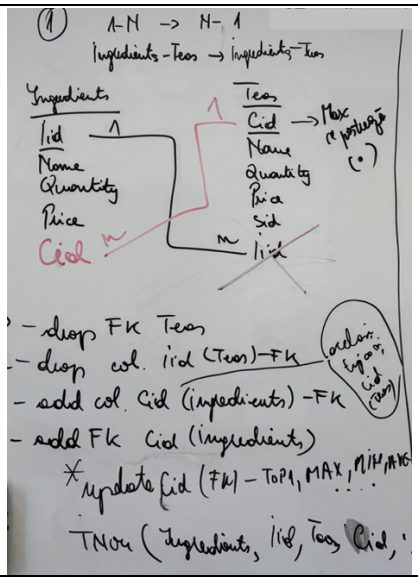


1:N	N:1
Iid – primary key in Ingredients (first table created)	- no changes
Iid – foreign key in Teas (second table created)	- Iid is removed as foreign key from Teas (*drop FK constraint* + *drop column*) - Store the maximum value of the primary key (Cid from Teas) corresponding to each primary key Iid from Ingredients, used *
	Cid is added in the Ingredients table as a foreign key (*add column* with the same type (and values) as the PK Cid from Teas + *add FK contraint* on this atribute in Ingredients)
	* make sure that the column *Cid - FK* from Ingredients is populated / updated with values (top 1, max, min, average, ...)

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Ingredients', Iid, 'Teas', Cid, 'relationship 1-N Ingredients – Teas was transformed into relationship N-1 Ingredients – Teas', '25.11.2025 8:51:00')

OR Blackboard drawing



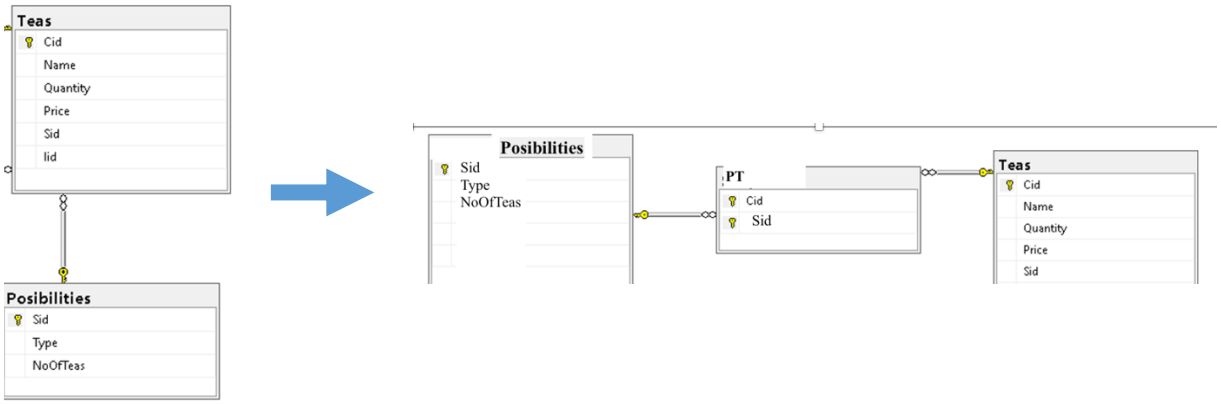
• * keep the max Cid (Teas) (for the corresponding Iid – Ingredients) • drop FK Teas (on Iid) • drop column Iid (table Teas, the FK) • add column Cid (in Ingredients – to set it after as FK) – with the same type as Cid (from Teas) • add FK on Cid (in Ingredients) • update the values of Cid – FK (in Ingredients) – top 1, max, min, avg, ... Insert into RemovedRelationships values ('Ingredients', 'Iid', 'Teas', 'Cid', 'relationship 1-N Ingredients – Teas was changed to N-1 Ingredients – Teas', '25.11.2025 8:51:00')	

Generic:
Consider a table A in 1:N relationship with table B. Table A is connected with multiple rows from table B. To take the relationship to N:1:

- The foreign key from table B is moved into table A
- The tables structure is modified such that each row from table A will reference a row from table B
- If a record reference multiple (old) records, is kept the foreign key with the maximum value (due to the first rule from the requirement)
- The records for which the references were deleted, will be inserted in RemovedRelationships table.

2. transformați o relație **1:N** în **M:N**;

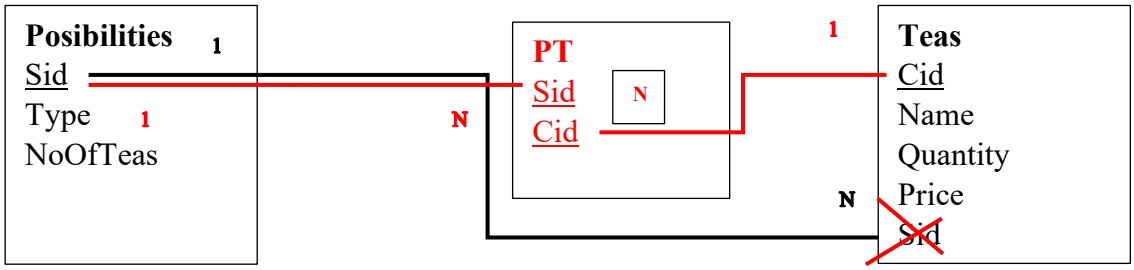
1:N – Posibilities – Teas (a posibility describes multiple teas and a tea is described by only one posibility) will be transformed in **M:N – Posibilities – (PT) – Teas** (multiple posibilities describe multiple teas)

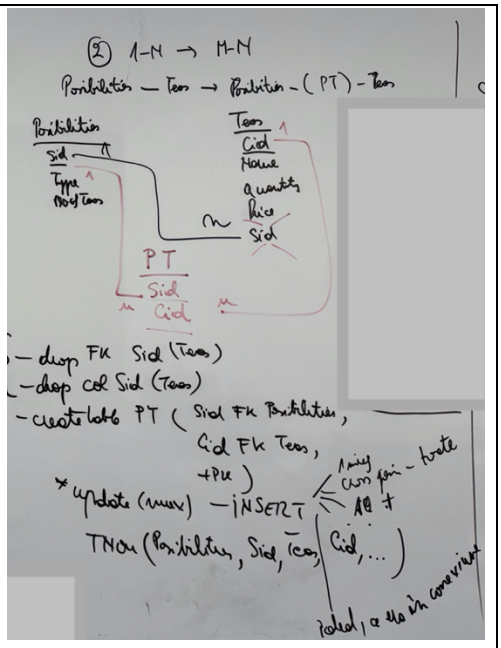


1:N	M:N
Sid – primary key in Posibilities (first table created)	- no changes
Sid – foreign key in Teas (second table created)	- Sid is removed as foreign key from Teas (*drop FK constraint* + *drop column*) - Store the maximum value of the primary key (Cid from Teas) corresponding to each primary key Sid from Posibilities, used *(if posible)
	Create table PT – intermediate table that connects Posibilities with Teas • Sid – will have the same type (and values) as Sid (from Posibilities) and it will reference it – FK • Cid – will have the same type (and values) as Cid (from Teas) and it will reference it – FK • Establish a PK – a solution is the pair (Sid, Cid)
	* make sure that the columns *Cid* - *Sid* – FK from PT are populated / updated with values (top 1, max, min, average, ...)

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Posibilities', Sid, 'Teas', Cid, 'relationship 1-N Posibilities – Teas was transformed into relationship M-N Posibilities – Teas', '25.11.2025 10:53:00')

OR Blackboard drawing



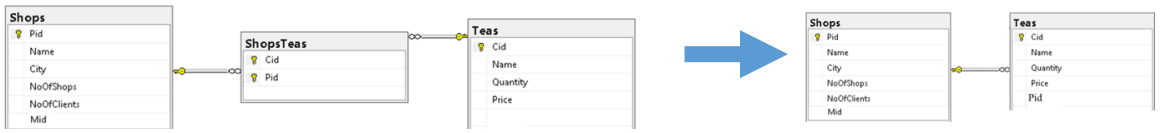
• * keep the max Cid (Teas) (for the corresponding Sid – Posibilities)
• drop FK Teas (on Sid)
• drop column Sid (table Teas, the FK)
• create table PT – intermediate table that connects Posibilities with Teas
  ○ Sid – will have the same type (and values) as Sid (from Posibilities) and it will reference it – FK
  ○ Cid – will have the same type (and values) as Cid (from Teas) and it will reference it – FK
  ○ Establish a PK – a solution is the pair (Sid, Cid)
 	
• Insert values in table PT (at least 10) – simple insert, cross join (all posible combinations), combination of the values of Cid – FK (Teas) and Sid – FK (Posibilities) – top 1, max, min, avg, ...
 	
Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Posibilities', Sid, 'Teas', Cid, 'relationship 1-N Posibilities – Teas was transformed into relationship M-N Posibilities – Teas', '25.11.2025 10:53:00')
	

Generic:
Consider a table A in 1:N relationship with table B. Table A is connected with multiple rows from table B. To take the relationship to M:N:

- Create an intermediate table that ha sat least 2 columns that are foreign keys referencing the two tables considered initially (A and B)
- The foreign key from the N table is eliminated
- Multiple records are added in the intermediate tablet o reflect the multiplicity M:N (at least 10 records)

3. transformați o relație **M:N** în **1:N**;

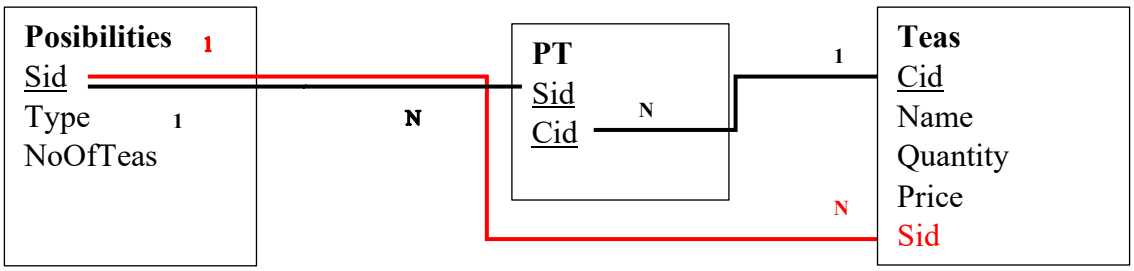
M:N – Shops – (ShopsTeas) – Teas (multiple shops sell multiple teas) will be transformed in **1:N – Shops – Teas** (a shop can sell multiple teas, and a tea can be sold by Only one shop)
* if the relationship considered is **M:N – Teas – (ShopsTeas) – Shops** will become **1:N – Teas – Shops**
* any of these 2 posible way is correct; it depends only about how the relationship is provided.



M:N	1:N
Pid – primary key in Shops	- no changes
Cid – primary key in Teas	- no changes
	- remove / drop intermediate table – ShopsTeas - add column Pid (in table Teas) - as a foreign key (*add column* with the same type (and values) as the PK Pid from Shops + *add FK contraint* on this atribute in Teas)
	* make sure that the column *Pid* – FK from Teas is populated / updated with values (top 1, max, min, average, ...)

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Shops', Sid, 'Teas', Cid, 'relationship M-N Shops – Teas was transformed into relationship 1-N Shops – Teas', '25.11.2025 17:23:00')

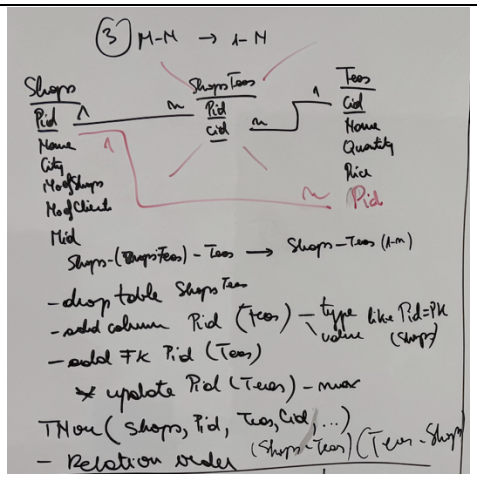
OR Blackboard drawing



- drop table ShopsTeas (columns with FK)
- add column *Pid* in Teas (with the same type (and values) as Pid (from Shops) and then reference it – FK)
- add FK contraint on Pid in Teas

- update the values of Pid – FK (in Teas) – top 1, max, min, avg, ...

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Shops', Sid, 'Teas', Cid, 'relationship M-N Shops – Teas was transformed into relationship 1-N Shops – Teas', '25.11.2025 17:23:00')

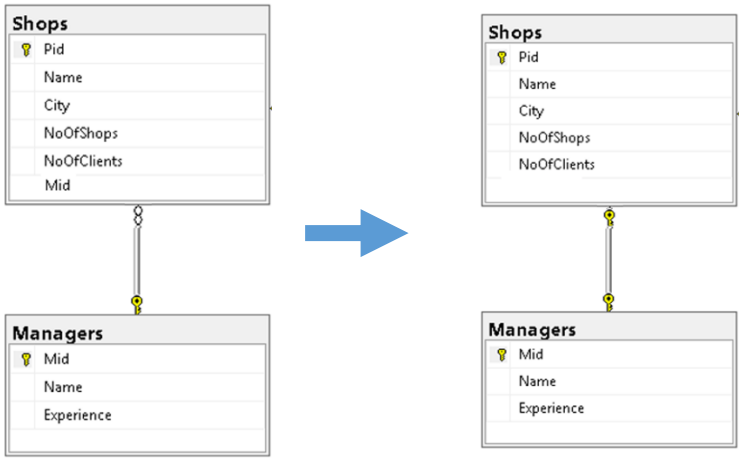


Generic:
Consider a table A in M:N relationship with table B. Table A and B are connected in multiple rows by the itnermediate table C. To take the relationship to 1:N:

- Choose the table that will keep the foreign key
- Remove / Drop the intermediate table, such that the relationship can be changed into 1:N
- If a record from the side 1 reference multiple records in the side N of the relationship, keep the maximum corresponding value (PK value) (due to the first rule)

4. transformați o relație **1:N** în **1:1**.

1:N – **Managers – Shops** (a manager can manage multiple shops and a shop can be managed by only one manager) will be transformed in **1:1 – Managers – Shops** (a manager is managing one shop and a shop is managed by one manager)



For example,

Managers

Mid	Name	Experience
1	A	10
2	B	8
3	C	12

Shops

Pid	Name	City	NoOfShops	NoOfClients	**Mid**
1	S1	Alba	2	12	**1**
2	S2	Cluj	3	10	**1**
3	S3	Pitesti	1	20	**1**
4	S4	Cluj	3	13	**2**
5	S5	Deva	2	18	**2**
6	S6	Dolj	1	24	**2**
7	S7	Alba	3	10	**2**
8	S8	Cluj	1	9	**3**
9	S9	Aiud	2	20	**3**

So, we keep only Shops with maximum Pid for every single manager (due to the first rule).
* even if the shops won't dissapear in real-life if They don't have temporary a manager

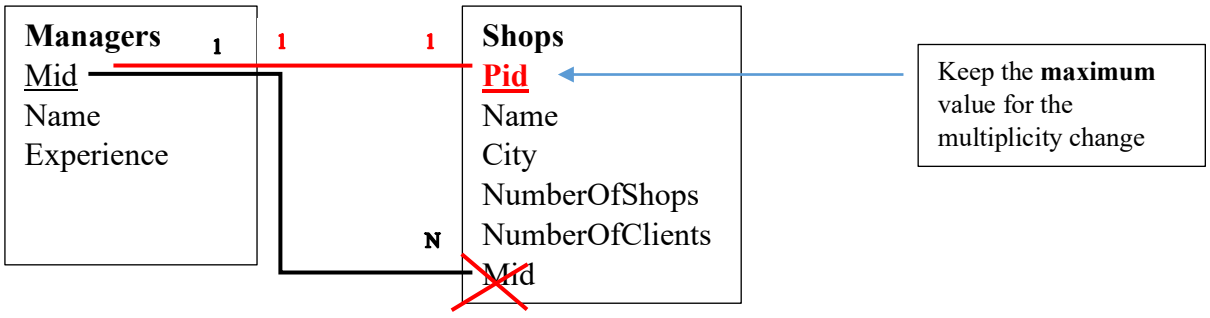
This multiplicity change can be performed in various ways.

Version 1 - FK

1:N	1:1
Mid – primary key in Managers (first table created)	- no changes
Pid – primary key in Shops (second table created) Mid – foreign key in Shops	- Mid is removed as foreign key from Shops (*drop FK constraint + drop column*) - Store the maximum value of the primary key (Pid from Shops) corresponding to each primary key Mid from Managers, used *
	- add to Pid (Shops) FK constraint (check to have the same type and values with Mid from Managers) * if Pid and Mid have diferent values - update (if possible) * if there are multiple Shops than Managers, add / insert extra managers and connect them with the shops (you can use a cursor ☺)
	* make sure that the column *Pid - FK* from Shops is populated / updated with values (top 1, max, min, average, ...)

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Managers', Mid, 'Shops', Pid, 'relationship 1-N Managers – Shops was transformed into relationship 1-1 Managers – Shops', '25.11.2025 19:36:00')

OR Blackboard drawing



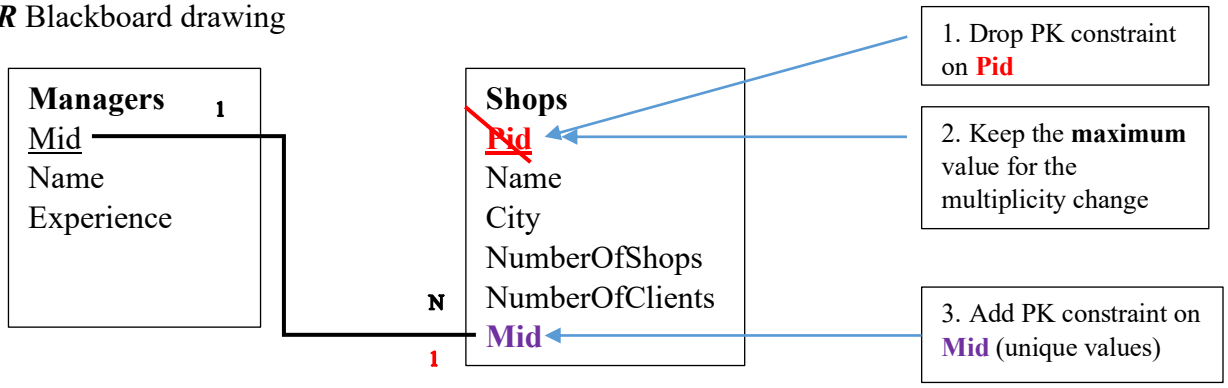
Version 2 – PK

1:N	1:1
Mid – primary key in Managers (first table created)	- no changes
Pid – primary key in Shops (second table created) Mid – foreign key in Shops	- Pid is removed as primary key from Shops * (*drop PK constraint)*

	* store the maximum Pid (and the corresponding record) to be able to keep only unique values on Mid
	- add to Mid (Shops) PK constraint (check to have unique values, that will connect with Mid from Managers**) ** otherwise, delete the duplicates - optional, can be removed the Pid column

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Managers', Mid, 'Shops', Pid, 'relationship 1-N Managers – Shops was transformed into relationship 1-1 Managers – Shops', '25.11.2025 19:36:00')

OR Blackboard drawing



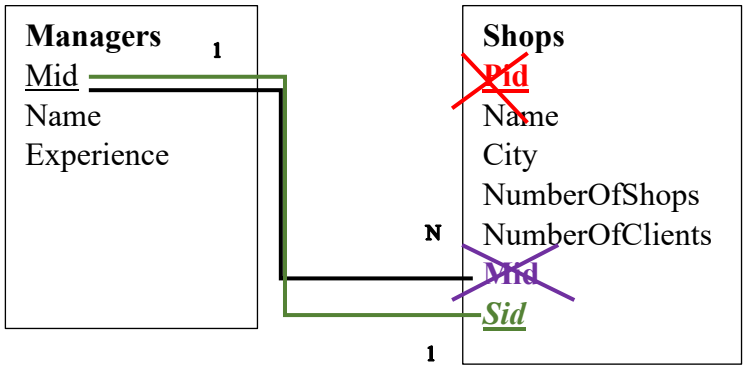
Version 3 – new column + (PK+FK)

1:N	1:1
Mid – primary key in Managers (first table created)	- no changes
Pid – primary key in Shops (second table created) Mid – foreign key in Shops	- add a new column **Sid** in Shops with the same type (and values) as Mid from Managers - update Sid with unique values as follows: * update Sid values with the ones of Mid (in Shops) – if there is the same number of values for Mid (Shops) and Mid (Managers) ** update Sid values with the ones corresponding to the maximum Pid (in Shops)
	- Mid is removed as foreign key from Shops (*drop FK constraint + column)* - Pid is removed as primary key from Shops (*drop PK constraint + column)*
	- Sid is set as PK and FK in Shops

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Managers', Mid,

'Shops', Pid, 'relationship 1-N Managers – Shops was transformed into relationship 1-1 Managers – Shops', '25.11.2025 19:36:00')

OR Blackboard drawing

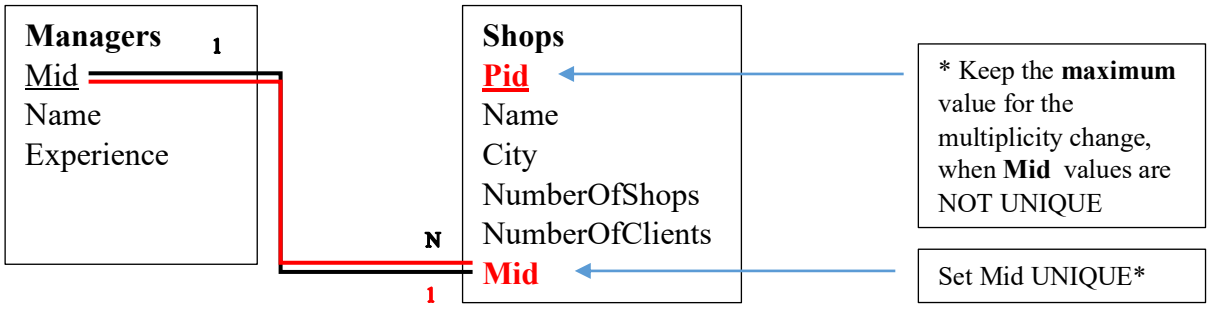


Version 4 – FK unique

1:N	1:1
Mid – primary key in Managers (first table created)	- no changes
Pid – primary key in Shops (second table created) Mid – foreign key in Shops	- add a UNIQUE constraint to Mid (Shops) * * make sure the Mid values are distinct; otherwise, keep only the records with the maximum Pid (due to the first rule), and delete the others

Insert into RemovedRelationships(LeftTableName, PKLeftTableName, RightTableName, PKRightTableName, MultiplicityDescription, MultiplicityDate) values ('Managers', Mid, 'Shops', Pid, 'relationship 1-N Managers – Shops was transformed into relationship 1-1 Managers – Shops', '25.11.2025 19:36:00')

OR Blackboard drawing



Generic:
Consider a table A in 1:N relationship with table B. Table A is connected with multiple rows from table B. To take the relationship to 1:1:

- Set the UNIQUE contraint on the foreign key in table B.
- Make sure that the values are unique on the foreign key in table B. Otherwise, for each foreign key value keep only the (primary) key with the maximum value, and delete the rest of records.

Short Steps:

- Choose 3 relationships 1:N and 1 relationship M:N from your database (create new one if need it).
- Insert at least 10 records per each table considered
- Create a new table to store multiplicity change
- Create **one stored procedure** to handle all the multiplicity changes (with **IF**)

OR

- Create **4 stored procedures** to handle each multiplicity change separately

OR

- **...**

Validation

Can be realized at various levels, such as:

- Tables existence
- Relationships existence
- Parameters of stored procedure
- Primary key constraint, foreign key constraint, …
- …

Feedback

Can be provided for any useful multiplicity modifications, such as:

- Multiplicity changes: initial, partial, final
- Steps in the process of changing multiplicity
- …

Generic short conclusion

Consider the following tables:

- A and B in relationship 1:N (table B contains the Foreign Key to table A)
- C and D in relationship M:N (intermediate table CD)

Multiplication Change	***Description***	***Example***
1:N into N:1	The foreign key from the N side is moved in 1 side of the relationship	Foreign key from B is moved in A
1:N into M:N	Create an intermediate table Remove the foreign key from the N side of the relationship	Intermediate table AB
M:N into 1:N	Keep the foreign key in the N side of the relationship Drop the intermediate table	Create foreign key in D Drop table CD

1:N into 1:1	To the foreign key from N side is added the UNIQUE constraint	The key from B to A is set to unique

For every step:

- If a record reference multiple records, is kept the maximum corresponding value, and the rest are deleted.
- The multiplication changed is salved into the table RemovedRelationships.