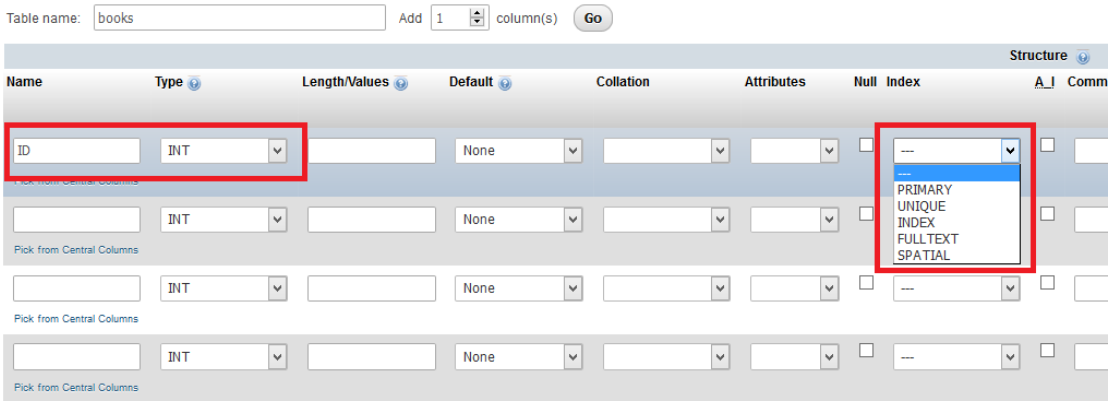
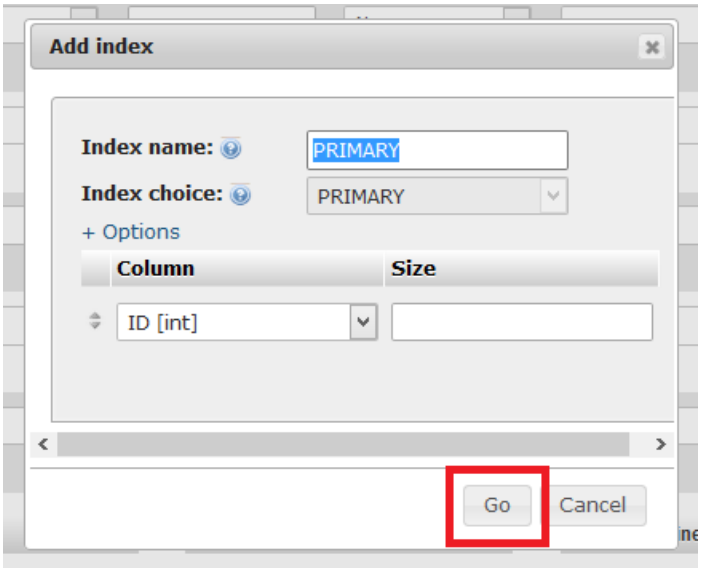
Δημιουργία Πινάκων

Για τη δημιουργία πινάκων στο περιβάλλον **phpMyAdmin** μπορούμε είτε να χρησιμοποιήσουμε τη φόρμα δημιουργίας πίνακα, είτε να εκτελέσουμε ένα ερώτημα SQL.

Στη παρακάτω εικόνα φαίνεται μια κενή φόρμα δημιουργίας νέου πίνακα. Στο πρώτο πεδίο έχει εισαχθεί το κείμενο «**ID**» για το όνομα του πρώτου πεδίου, με τύπο **INT** (**integer**). Αυτό το πεδίο προορίζεται για **πρωτεύον κλειδί** (**primary key**). Γι' αυτό το λόγο ενεργοποιούμε την επιλογή «**PRIMARY**» στη στήλη «**Index**».

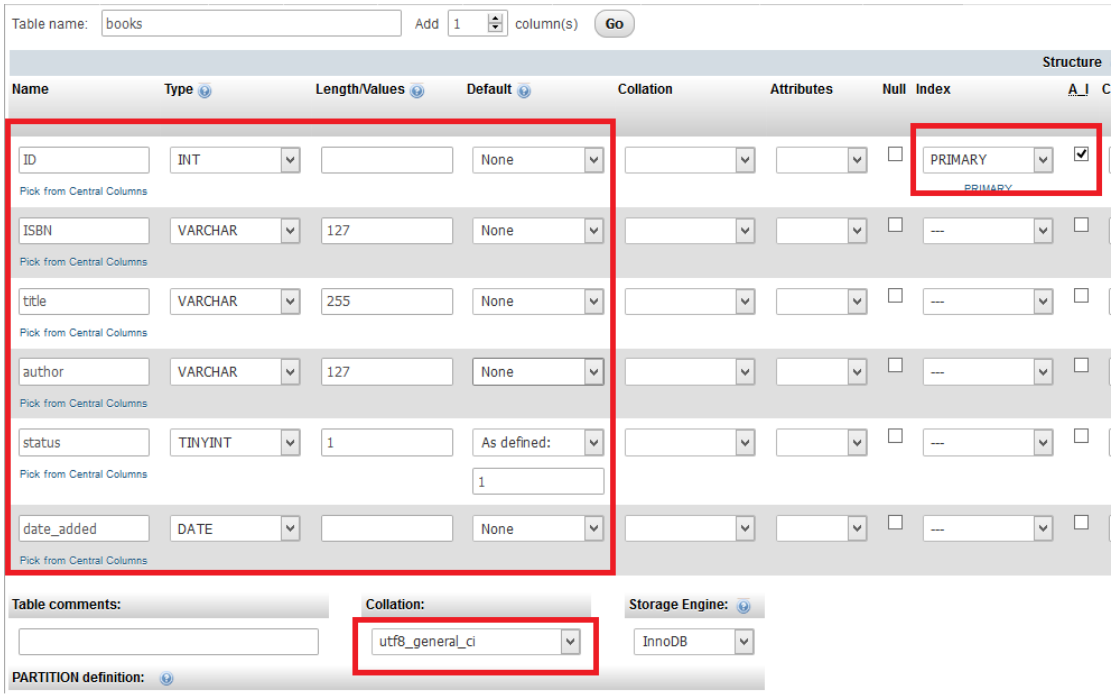


Κατά την ενεργοποίηση της ιδιότητας αυτής ενδέχεται να εμφανιστεί ένα μήνυμα σαν αυτό της παρακάτω εικόνας. Απλά επιβεβαιώστε πατώντας OK.



Στη συνέχεια ενεργοποιούμε την **αυτόματη αύξηση** (**Auto Increment** – **A_I**) για το πεδίο «**ID**». Αυτή η επιλογή αναθέτει στη βάση την αυτόματη αύξηση του πρωτεύοντος κλειδιού.

Στην παρακάτω εικόνα φαίνονται συμπληρωμένα τα βασικά πεδία ενός πίνακα για την αποθήκευση των βιβλίων μιας βιβλιοθήκης. Επίσης αναθέτουμε στη βάση τη κωδικοποίηση «**utf8_general_ci**». Αφού συμπληρωθούν τα παρακάτω πεδία, πατάμε «αποθήκευση» για να δημιουργηθεί ο πίνακας.



Σε αυτό το σημείο πρέπει να πούμε ότι συνήθως προηγείται η πλήρης σχεδίαση της βάσης σύμφωνα με τις προδιαγραφές της εφαρμογής. Ακολουθεί η κατασκευή της βάσης, η οποία μπορεί βεβαίως να τροποποιηθεί όταν αυτό κρίνεται απαραίτητο.

Κατά την συμπλήρωση της παραπάνω φόρμας μας δίνεται η δυνατότητα να ελέγξουμε το ερώτημα SQL που προκύπτει από αυτήν. Αυτό μπορεί να γίνει από το κουμπί «**Προεπισκόπηση SQL**». Για τη συγκεκριμένη φόρμα το ερώτημα που δημιουργείται είναι το εξής

```
CREATE TABLE `books` (
        `ID` INT NOT NULL AUTO_INCREMENT ,
        `ISBN` VARCHAR (127) NOT NULL ,
        `title` VARCHAR(255) NOT NULL ,
        `author` VARCHAR(127) NOT NULL ,
        `price` decimal(15,2) NOT NULL ,
        `status` TINYINT(1) NOT NULL DEFAULT '1' ,
        `date_added` DATE NOT NULL ,
        PRIMARY KEY (`ID`)
) ENGINE = InnoDB CHARSET=utf8 COLLATE utf8_general_ci;
```

Αντί λοιπόν να δημιουργήσουμε το πίνακα με τη χρήση της φόρμας, θα μπορούσαμε να εκτελέσουμε αυτό το ερώτημα από το περιβάλλον **phpMyAdmin** και τη καρτέλα «**SQL**».

Σε περίπτωση που δημιουργούσαμε τον πίνακα χωρίς ανάθεση πρωτεύοντος κλειδιού, αυτό θα μπορούσε να ορισθεί μετά. Τότε θα μπορούσαμε να εκτελέσουμε τρία ερωτήματα. Ένα για τη δημιουργία του πίνακα και άλλα δυο για τη τροποποίησή του. Τα ερωτήματα αυτά παρουσιάζονται παρακάτω. Συνήθως αυτή είναι η μορφή των ερωτημάτων που παράγονται όταν εξάγουμε τη βάση μέσω του **phpMyAdmin**.

```
CREATE TABLE `books` (
        `ID` int(11) NOT NULL,
        `ISBN` varchar(127) NOT NULL,
        `title` varchar(255) NOT NULL,
        `author` varchar(127) NOT NULL,
        `price` decimal(15,2) NOT NULL ,
        `status` tinyint(1) NOT NULL DEFAULT '1',
        `date_added` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `books` ADD PRIMARY KEY (`ID`);
ALTER TABLE `books` MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT;
```

Τροποποίηση Πινάκων

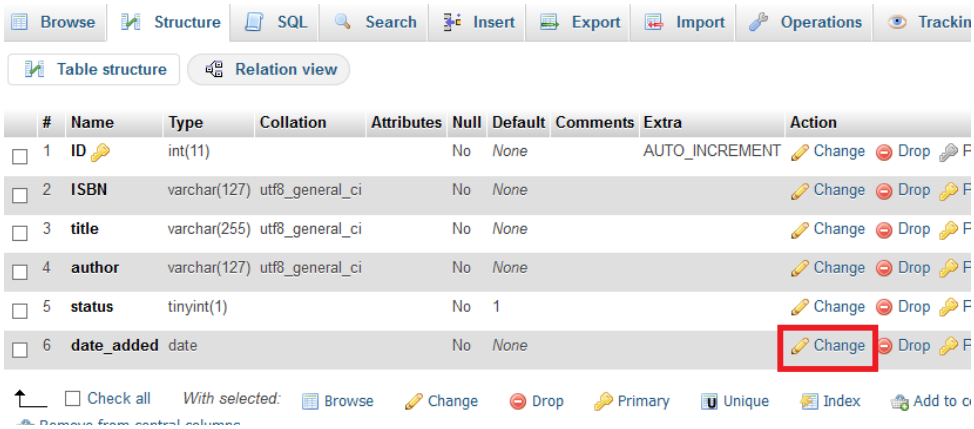
Όπως είπαμε η δομή του πίνακα μπορεί να χρειαστεί να αλλάξει μετά τη δημιουργία του. Αυτό μπορεί κριθεί απαραίτητο αν αλλάξουν οι προδιαγραφές ή αν κάτι δεν έχει γίνει σωστά κατά τη δημιουργία του πίνακα.

Όπως και κατά τη δημιουργία του πίνακα αυτό μπορεί να γίνει με το γραφικό περιβάλλον ή με εντολές της SQL. Ένα παράδειγμα τέτοιων εντολών βλέπουμε στο τέλος της προηγούμενης ενότητας, όπου χρησιμοποιούνται δυο εντολές τροποποίησης για να οριστεί το κλειδί του πίνακα.

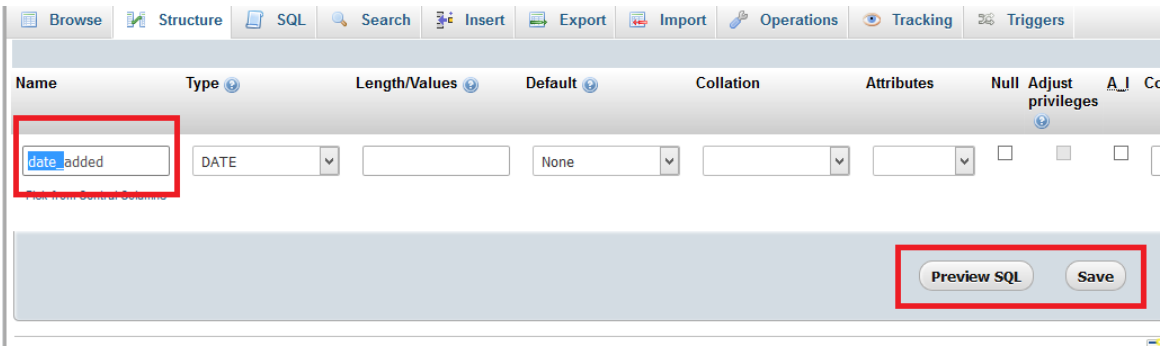
```
ALTER TABLE `books` ADD PRIMARY KEY (`ID`);
ALTER TABLE `books` MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT;
```

Με τον ίδιο τρόπο μπορεί να αλλάξει μια ιδιότητα για κάποιο πεδίο, να διαγραφεί ένα πεδίο ή να προστεθεί νέο.

Έστω για παράδειγμα ότι θέλουμε να αλλάξουμε το όνομα του πεδίου '**date_added**' και να το κάνουμε '**added'**. Από το γραφικό περιβάλλον, μπορούμε να πάμε στη δομή του πίνακα και να πατήσουμε το κουμπί '**change'.**



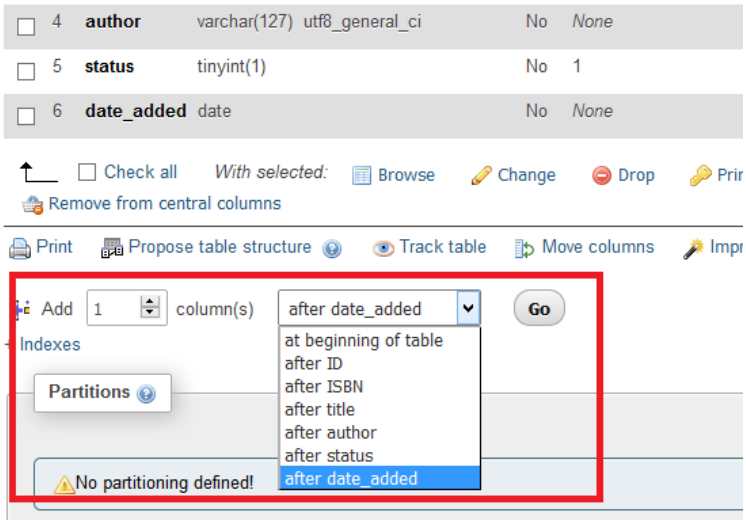
Στη φόρμα που θα ανοίξει μπορούμε να αλλάξουμε το όνομα ή κάποια άλλη ιδιότητα του πεδίου. Στη συνέχεια μπορούμε να δούμε ποιο είναι το ισοδύναμο ερώτημα που θα εφαρμόσει την αλλαγή, πατώντας το κουμπί «**προεπισκόπηση SQL**». Με το κουμπί «**Save**» εφαρμόζουμε τις αλλαγές.



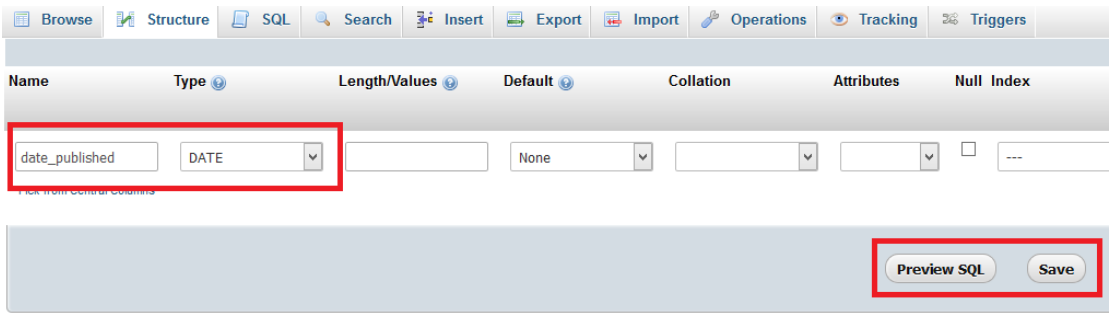
Το ισοδύναμο ερώτημα είναι το εξής:

```
ALTER TABLE `books` CHANGE `date_added` `added` DATE NOT NULL;
```

Στο κάτω μέρος της σελίδας της δομής του πίνακα δίνεται η δυνατότητα να προσθέσουμε νέα πεδία στον πίνακα. Απλά επιλέγουμε πόσα θέλουμε και που θα τοποθετηθούν σε σχέση με τα υπάρχοντα πεδία. Ας προσθέσουμε ένα ακόμα πεδίο «**date_published**» μετά από το πεδίο «**date_added**». Το πεδίο αυτό θα είναι επίσης τύπου **Date**. Επιλέγουμε λοιπόν προσθήκη ενός πεδίου μετά από το πεδίο «**date_added**» και πατάμε ΟΚ.



Στη φόρμα που ακολουθεί, συμπληρώνουμε τις ιδιότητες που θέλουμε για το νέο πεδίο και πατάμε «Save» για την εφαρμογή των αλλαγών



Το ισοδύναμο ερώτημα είναι το εξής:

```
ALTER TABLE `books` ADD `date_published` DATE NOT NULL AFTER `date_added`;
```

Αν θέλαμε να αφαιρέσουμε ένα πεδίο θα μπορούσαμε να πατήσουμε το κουμπί «**drop**» αντί για το κουμπί «**change**». Το ισοδύναμο ερώτημα είναι το εξής:

```
ALTER TABLE `books` DROP `date_published`;
```

Σημείωση: Η αλλαγή των ιδιοτήτων ενός πεδίου μπορεί να γίνει είτε με την «**CHANGE»** είτε με την «**MODIFY»**. Στη πρώτη πρέπει να αναφέρεις ξανά όλες τις ιδιότητες του πεδίου, ενώ στη δεύτερη μόνο το όνομα και τον τύπο και ύστερα τις αλλαγές. Έστω για παράδειγμα ότι θέλουμε να επιτρέψουμε τη τιμή **NULL** στο πεδίο **Author**.

```
ALTER TABLE `books` CHANGE `author` `author` VARCHAR(127) CHARACTER SET utf8
COLLATE utf8_general_ci NULL;


ALTER TABLE `books` MODIFY `author` VARCHAR(127) NULL;
```

Συσχετίσεις Πινάκων

Για να κάνουμε τα αρχικά παραδείγματα πιο απλά, ορίσαμε τον πίνακα books έτσι ώστε να ενσωματώνει όλες σχεδόν τις απαραίτητες πληροφορίες. Έτσι για παράδειγμα το όνομα του συγγραφέα και είναι ενσωματωμένο σε κάθε εγγραφή βιβλίου. Κάτι τέτοιο όμως οδηγεί σε **επανάληψη πληροφορίας** και δημιουργεί προβλήματα στην **ακεραιότητα** της βάσης.

Για παράδειγμα αν στη βάση υπάρχουν 3 βιβλία του **Stephen King** τότε σε κάθε ένα από αυτά θα αναφέρεται το όνομά του. Κάτι τέτοιο πρέπει να αποφεύγεται και προτείνεται η δημιουργία ενός δεύτερου πίνακα για την αποθήκευση της οντότητας «Συγγραφέας»(author). Η νέα αυτή οντότητα συσχετίζεται στη συνέχεια με την οντότητα του βιβλίου, δημιουργώντας έτσι μια σχέση ανάμεσα στους δυο πίνακες. Γι' αυτό το λόγο άλλωστε οι βάσεις που μας απασχολούν ονομάζονται **σχεσιακές**.

Ανάλογα με το μοντέλο της σχέσης των πινάκων διακρίνουμε τις σχέσεις **ένα προς ένα** (1:1), **ένα προς πολλά** (1:N) και **πολλά προς πολλά** (N:M).

Σχέση ένα προς πολλά (1:N)

Στη περίπτωση της συσχέτισης του πίνακα των βιβλίων με ένα νέο πίνακα συγγραφέων, η σχέση που προκύπτει είναι **ένα προς πολλά** (1:N), αφού ένας συγγραφέας μπορεί να συσχετιστεί με πολλά βιβλία, αλλά ένα βιβλίο με έναν μόνο συγγραφέα.

Για να υλοποιηθεί αυτή η σχέση πρέπει πρώτα να δημιουργήσουμε το πίνακα των συγγραφέων. Μπορούμε να χρησιμοποιήσουμε το παρακάτω ερώτημα

```
CREATE TABLE `author` (
    `ID_auth` int(11) UNIQUE NOT NULL AUTO_INCREMENT,
    `name` varchar(127) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Στη συνέχεια θα πρέπει να αλλάξουμε το πίνακα των βιβλίων ώστε αντί να αποθηκεύουμε το όνομα του συγγραφέα να αποθηκεύουμε το **id** του από τον πίνακα **author**. Η τροποποίηση μπορεί να γίνει με ένα ερώτημα **ALTER TABLE** ως εξής:

```
ALTER TABLE `books` MODIFY `author` INT(11)  NOT NULL;
```

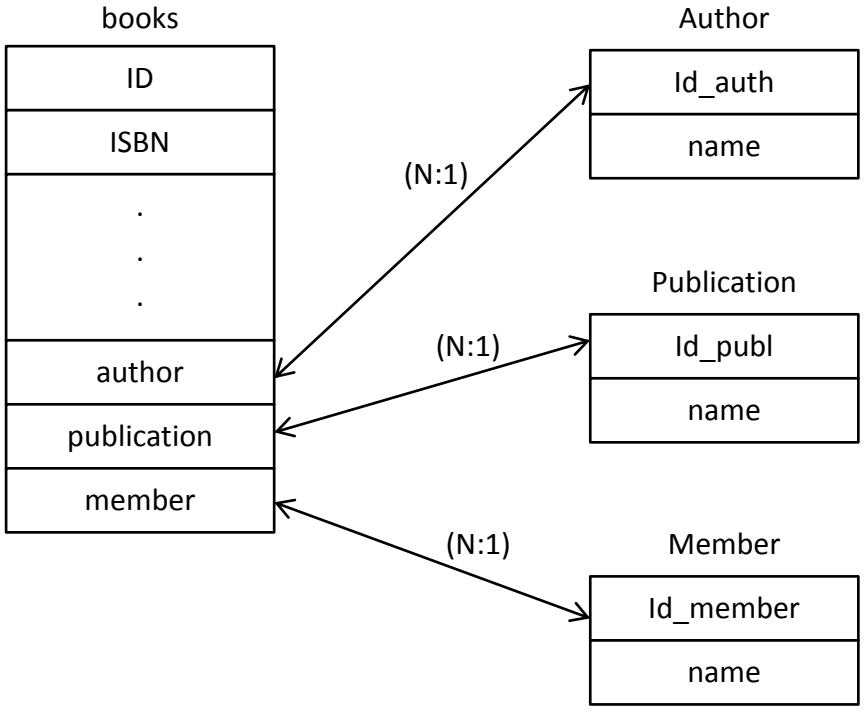
Για τον ίδιο σκοπό που δημιουργήθηκε ο πίνακας των συγγραφέων θα δημιουργήσουμε και ένα πίνακα για την αποθήκευση των εκδοτικών οίκων, «**publication**» και θα προσθέσουμε ένα ακόμα πεδίο στον πίνακα books για να δημιουργηθεί η σχέση.

```
CREATE TABLE `publication` (
        `ID_publ` int(11) UNIQUE NOT NULL AUTO_INCREMENT,
        `name` varchar(127) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
ALTER TABLE `books` ADD `publication` INT(11) NOT NULL;
```

ΣΗΜΕΙΩΣΗ: Σε αυτό το σημείο να αναφέρουμε ότι οι σχέσεις μεταξύ πινάκων ορίζονται πολλές φορές στις σχεσιακές βάσεις με τη δήλωση **ξένου κλειδιού** (**Foreign Key**). Αυτή η μέθοδος δεν υποστηρίζεται από όλες της μηχανές της MySQL. Όπου όμως υποστηρίζεται (π.χ. στην **InnoDB**) μπορεί να βοηθήσει ώστε η βάση να διαχειρίζεται μόνη της κάποιες από τις λειτουργίες και, δεδομένης της σύνδεσης των πινάκων, να εγγυάται την ακεραιτότητα της βάσης. Στο δικό μας παράδειγμα δεν χρησιμοποιήθηκε μια τέτοια δήλωση ξένου κλειδιού.

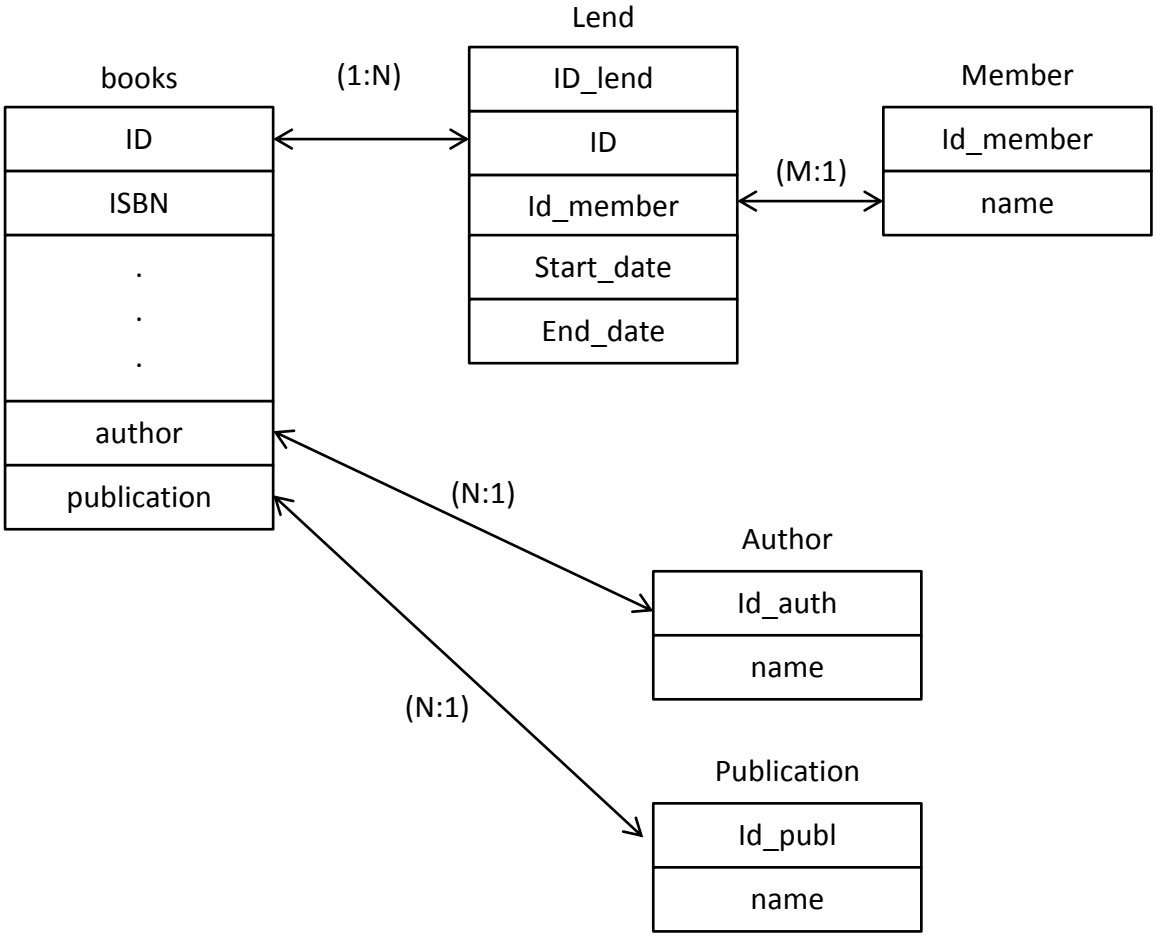
Σχέση πολλά προς πολλά (M:N)

Οι σχέσεις που υλοποιήθηκαν μέχρι τώρα είναι και οι δυο τύπου «**ένα προς πολλά** (1:N)». Θα υλοποιήσουμε ακόμη μια σχέση «**πολλά προς πολλά** (N:M)». Ας υποθέσουμε ότι θέλουμε να αποθηκεύουμε τη πληροφορία για το ποια βιβλία έχουν δανειστεί τα μέλη της βιβλιοθήκης (ή αλλιώς ποιος έχει δανειστεί το κάθε βιβλίο). Σε πρώτη φάση γίνεται σαφές ότι χρειαζόμαστε ένα πίνακα για την οντότητα **μέλος** (**member**) και ένα επιπλέον πεδίο "**member**" στο πίνακα των βιβλίων. Ο πίνακας των μελών θα μπορούσε να έχει πολύ περισσότερα πεδία αλλά θα του δώσουμε μόνο δυο πεδία για ευκολία.



Ο πίνακας **Member** συνδέεται με το πίνακα **books** μέσω του κλειδιού. Έτσι, όπως και στις άλλες δυο σχέσεις, μπορούμε σε ένα βιβλίο να βάλουμε στο πεδίο **member** το **ID** του μέλους ώστε να επιτευχθεί η δήλωση του ποιος έχει δανειστεί το βιβλίο. Πάλι όμως έχουμε μια σχέση (1:N) και μάλιστα η μόνη πληροφορία που έχουμε αποθηκεύσει είναι το «**ποιος έχει δανειστεί το βιβλίο τώρα**». Αν θέλαμε να έχουμε ένα ιστορικό του ποιος έχει δανειστεί το βιβλίο κατά το παρελθόν τότε οι πληροφορίες θα ήταν περισσότερες. Μάλιστα δεν θα γνωρίζαμε πόσες ακριβώς φορές το έχει δανειστεί κάποιος και άρα δεν θα μπορούσαμε να ενσωματώσουμε αυτή τη πληροφορία σε κάποιον από τους πίνακες **books** και **members** βάζοντας ένα επιπλέον πεδίο.

Για την αντιμετώπιση αυτής της κατάστασης χρειαζόμαστε έναν επιπλέον πίνακα για την αποθήκευση αυτής της πληροφορίας. Σε αυτό το πίνακα (έστω **lend**) θα τοποθετούσαμε ζευγάρια τιμών βιβλίο-μέλος για να δηλώσουμε ότι ένα μέλος δανείστηκε ένα βιβλίο. Θα μπορούσαμε ακόμα να αποθηκεύσουμε και την ημερομηνία δανεισμού και επιστροφής. Το προηγούμενο μοντέλο λοιπόν θα άλλαζε στο εξής:



Με τη βοήθεια του πίνακα **lend** ουσιαστικά υλοποιούμε τη σχέση «**πολλά προς πολλά (N:M)**» που συνδέει τους δυο πίνακες, **members** και **books**.

Ακολουθούν τα ερωτήματα SQL που θα χρησιμοποιούσαμε για τη δημιουργία των πινάκων.

```
CREATE TABLE ` member ` (
        `ID_member` INT (11) UNIQUE NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(127) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Lend` (
        `ID_lend` INT(11) UNIQUE NOT NULL AUTO_INCREMENT,
        `ID` INT(11) NOT NULL,
        `ID_member` INT(11) NOT NULL,
        `start_date` DATE NOT NULL,
        `end_date` DATE NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```