# Εισαγωγή στην SQL

Για να μπορέσουμε να δημιουργήσουμε και να διαχειριστούμε μια βάση δεδομένων, μπορούμε να χρησιμοποιήσουμε ειδικές γλώσσες προγραμματισμού, τις λεγόμενες *γλώσσες ερωταπαντήσεων (query languages)*. Είναι γλώσσες μη διαδικαστικές, τέταρτης γενιάς ($4^{th}$ generation languages). Εμείς απλά διατυπώνουμε με απλές και κατανοητές εντολές το τι πληροφορίες ζητάμε και το ΣΔΒΔ (Σύστημα Διαχείρισης Βάσεων Δεδομένων) αναλαμβάνει να μας απαντήσει. Η *SQL (Structured Query Language, δηλ. Δομημένη Γλώσσα Ερωταπαντήσεων)* είναι σήμερα η πιο δημοφιλής και πιο διαδεδομένη γλώσσα ανάπτυξης και διαχείρισης σχεσιακών βάσεων δεδομένων.

Η SQL αποτελείται από εντολές με τα ορίσματά τους, τις οποίες μπορούμε να χρησιμοποιήσουμε με συγκεκριμένους κανόνες σύνταξης για να πάρουμε τα αποτελέσματα που θέλουμε. Με την SQL μπορούμε να δημιουργήσουμε μια βάση δεδομένων και τους πίνακές της με τα αντίστοιχα πεδία, να καταχωρήσουμε δεδομένα στους πίνακες, να τροποποιήσουμε και να διαγράψουμε τα δεδομένα αυτά, να αλλάξουμε τη δομή των πινάκων με προσθήκη και διαγραφή πεδίων και να εμφανίσουμε πληροφορίες (συνδυασμούς από δεδομένα).

Η SQL έχει δύο τμήματα :

- Τη *Γλώσσα Ορισμού Δεδομένων (DDL, Data Definition Language)*, η οποία περιέχει τις απαραίτητες εντολές για τον ορισμό και την τροποποίηση του σχεσιακού σχήματος καθώς και για τη δημιουργία, την τροποποίηση και τη διαγραφή σχέσεων. Περιέχει ακόμη τις εντολές δημιουργίας και επεξεργασίας όψεων και ορισμού περιορισμών ακεραιότητας.
- Τη *Γλώσσα Χειρισμού Δεδομένων (DML, Data Manipulation Language)*, η οποία περιέχει τις απαραίτητες εντολές για την εμφάνιση (αναζήτηση) δεδομένων καθώς και για την καταχώρηση, τροποποίηση και διαγραφή των εγγραφών (πλειάδων) μιας σχέσης.
- Τέλος, περιέχει εντολές για τον ορισμό και την επεξεργασία *συναλλαγών (transactions)*.

# Οι Τύποι Δεδομένων της SQL

Η SQL υποστηρίζει τους εξής τύπους δεδομένων για τα πεδία μιας σχέσης :

- *char(n)*, ένα αλφαριθμητικό (string) με n ακριβώς χαρακτήρες.
- *varchar(n)*, ένα αλφαριθμητικό (string) με μεταβλητό μήκος και με n το πολύ χαρακτήρες.
- *int*, ακέραιος αριθμός.
- *smallint*, ακέραιος αριθμός με μικρές τιμές.

- *Numeric(p, d)*, αριθμός με p ψηφία, από τα οποία τα d είναι δεκαδικά.
- *real*, αριθμός κινητής υποδιαστολής απλής ακρίβειας.
- *double precision*, αριθμός κινητής υποδιαστολής διπλής ακρίβειας.
- *float(n)*, αριθμός κινητής υποδιαστολής με ακρίβεια n ψηφίων.
- *date*, ημερομηνία (ημέρα, μήνας, έτος).
- *time*, ώρα (ώρα, λεπτά, δευτερόλεπτα).

Μπορούμε να δημιουργήσουμε και δικούς μας τύπους δεδομένων με την εντολή *create domain*, ως εξής :

*Create domain Name char(20);*

Ο νέος τύπος δεδομένων Name θα μπορεί να χρησιμοποιηθεί εφεξής όπως και οι υπόλοιποι τύποι δεδομένων της SQL. Απλά θα περιέχει strings με μήκος 20 χαρακτήρων.

# Δημιουργία, Τροποποίηση και Διαγραφή Σχέσεων

Για να δημιουργήσουμε μια σχέση (πίνακα) χρησιμοποιούμε την εντολή *create table*. Για παράδειγμα, για να δημιουργήσουμε τη σχέση Αθλητής δίνουμε την εξής εντολή :

```
Create table ΑΘΛΗΤΗΣ
    (Κωδικός_Αθλητή     integer not null,
    Επώνυμο             char(20) not null,
    Όνομα               char(15),
    Ρεκόρ               numeric(4, 2),
    Ημνία_Γέννησης      date,
    Κωδικός_Συλλόγου    integer not null,
    primary key (Κωδικός_Αθλητή),
    check(Κωδικός_Αθλητή >= 100));
```

Μετά την εντολή create table γράφουμε το όνομα της σχέσης (πίνακα) και ακολουθεί μια παρένθεση που περιέχει τα ονόματα των πεδίων με τους τύπους δεδομένων τους. Η δήλωση *not null* σημαίνει ότι το συγκεκριμένο πεδίο θα πρέπει να έχει οπωσδήποτε κάποια τιμή. Προσοχή : άλλο πράγμα είναι η τιμή 0 ή η τιμή του κενού χαρακτήρα (space), που είναι κάποια τιμή, και άλλο πράγμα είναι η μη ύπαρξη τιμής.

Με τη δήλωση *primary key* ορίζουμε το πεδίο κλειδί (πρωτεύον κλειδί) της σχέσης. Αν, αργότερα κατά την εισαγωγή τιμών, δώσουμε μια τιμή null στο

πρωτεύον κλειδί ή μια τιμή που ήδη υπάρχει σε κάποια άλλη πλειάδα, τότε θα εμφανισθεί ένα μήνυμα λάθους.

Με τον όρο *check* μπορούμε να δηλώσουμε μια συνθήκη για την περιοχή των τιμών ενός πεδίου. Είδαμε τη χρήση του όρου check για να ελέγξει αν η τιμή ενός κωδικού είναι μεγαλύτερη ή ίση από 100. Μπορούμε να χρησιμοποιήσουμε τον όρο check και για να ελέγξουμε αν η τιμή ενός πεδίου ανήκει σ' ένα προκαθορισμένο σύνολο τιμών, ως εξής :

***check** (Νομός **in** ('Γρεβενών', 'Καστοριάς', 'Κοζάνης', 'Φλώρινας'))*

Εδώ ελέγχουμε το αν η τιμή του πεδίου Νομός θα ανήκει σε μία από τέσσερις συγκεκριμένες τιμές.

Για να τροποποιήσουμε μια σχέση και να προσθέσουμε ένα πεδίο, δίνουμε την εντολή *alter table* και το όρισμα *add*, ως εξής :

***Alter table** ΑΘΛΗΤΗΣ **add** (Τηλέφωνο char(10));*

Τα καινούργια πεδία που προστίθενται σε μια σχέση, έχουν αρχικά τιμές ίσες με null. Για να τροποποιήσουμε μια σχέση και να διαγράψουμε ένα πεδίο, δίνουμε την εντολή ***alter table*** και το όρισμα *drop*, ως εξής :

***Alter table** ΑΘΛΗΤΗΣ **drop** Τηλέφωνο;*

Για να διαγράψουμε τελείως μια σχέση από τη βάση δεδομένων στην οποία ανήκει, μαζί με τις τιμές των εγγραφών της, δίνουμε την εντολή *drop table*, ως εξής :

***Drop table** ΑΘΛΗΤΗΣ;*

Υπάρχει και η εντολή ***delete from** ΑΘΛΗΤΗΣ*, με την οποία μπορούμε να διαγράψουμε τις τιμές (περιεχόμενα, εγγραφές) μιας σχέσης, αλλά όχι την ίδια τη σχέση.

# Η Ακεραιότητα Αναφορών

Με τον όρο *ακεραιότητα αναφορών (referential integrity)* αναφερόμαστε στην ιδιότητα όπου οι τιμές ορισμένων πεδίων μιας σχέσης υπάρχουν και σε αντίστοιχα πεδία σε κάποια άλλη σχέση. Για παράδειγμα, αν έχουμε τις σχέσεις ΑΘΛΗΤΗΣ και ΑΓΩΝΑΣ, τότε η συσχέτιση μεταξύ τους υλοποιείται με την τρίτη σχέση ΣΥΜΜΕΤΟΧΗ, όπου εμφανίζεται το ποιοι αθλητές συμμετείχαν σε ποιους αγώνες και φυσικά τι επίδοση κάνανε.

Η ακεραιότητα αναφορών έρχεται να επιλύσει το πρόβλημα της διαγραφής μιας πλειάδας από τη σχέση ΑΓΩΝΑΣ, οπότε οι αθλητές που συμμετείχαν στον συγκεκριμένο αγώνα θα φαίνονται ξεκρέμαστοι, καθώς επίσης και το πρόβλημα της τροποποίησης του πεδίου κλειδιού (Κωδικός_Αγώνα) μιας πλειάδας από τη σχέση ΑΓΩΝΑΣ, οπότε και πάλι οι αθλητές που συμμετείχαν στον συγκεκριμένο αγώνα θα φαίνονται ξεκρέμαστοι.

Για να αποφύγουμε τέτοιες καταστάσεις, καταφεύγουμε στην έννοια του *ξένου κλειδιού (foreign key)*, όπου στη σχέση ΣΥΜΜΕΤΟΧΗ, το πεδίο Κωδικός_Αθλητή είναι ξένο κλειδί αλλά και πρωτεύον κλειδί στη σχέση

ΑΘΛΗΤΗΣ, ενώ το πεδίο Κωδικός_Αγώνα είναι ξένο κλειδί στη σχέση ΣΥΜΜΕΤΟΧΗ αλλά και πρωτεύον κλειδί στη σχέση ΑΓΩΝΑΣ.

Με τη δήλωση της ακεραιότητας αναφορών, αν διαγράψουμε μια πλειάδα ενός πεδίου που είναι πρωτεύον κλειδί σε κάποια σχέση, τότε θα διαγραφούν και όλες οι αντίστοιχες εγγραφές (πλειάδες) στη σχέση όπου το ίδιο πεδίο είναι ξένο κλειδί. Αυτό σημαίνει πρακτικά ότι αν διαγράψουμε έναν αθλητή από τη σχέση ΑΘΛΗΤΗΣ, τότε θα διαγραφούν και όλες οι συμμετοχές του σε αγώνες από τη σχέση ΣΥΜΜΕΤΟΧΗ.

Επίσης, αν τροποποιήσουμε την τιμή ενός πεδίου που είναι πρωτεύον κλειδί σε κάποια σχέση, τότε θα ενημερωθούν αυτόματα και όλες οι αντίστοιχες εγγραφές (πλειάδες) στη σχέση όπου το ίδιο πεδίο είναι ξένο κλειδί. Αυτό σημαίνει πρακτικά ότι αν τροποποιήσουμε τον κωδικό ενός αγώνα από τη σχέση ΑΓΩΝΑΣ, τότε θα ενημερωθούν αυτόματα και όλες οι συμμετοχές των αθλητών στον αγώνα αυτό που υπάρχουν στη σχέση ΣΥΜΜΕΤΟΧΗ.

Για να δηλώσουμε ένα ξένο κλειδί και την αναφορά του, χρησιμοποιούμε τους όρους *foreign key* και *references* όταν δημιουργούμε μια σχέση, ως εξής :

```
Create table ΣΥΜΜΕΤΟΧΗ
        (Κωδικός_Αθλητή     integer not null,
        Κωδικός_Αγώνα       integer not null,
        Επίδοση              integer,
        primary key(Κωδικός_Αθλητή, Κωδικός_Αγώνα),
        foreign key(Κωδικός_Αθλητή) references ΑΘΛΗΤΗΣ,
        foreign key(Κωδικός_Αγώνα) references ΑΓΩΝΑΣ);
```

Σύμφωνα με τον παραπάνω τρόπο δημιουργίας της σχέσης ΣΥΜΜΕΤΟΧΗ, αν επιχειρήσουμε να διαγράψουμε έναν αθλητή που έχει συμμετάσχει σε κάποιον αγώνα ή έναν αγώνα στον οποίον έχουν συμμετάσχει κάποιοι αθλητές, το σύστημα δεν θα μας αφήσει να το κάνουμε γιατί έχουμε παραβίαση της ακεραιότητας αναφοράς. Το ίδιο πρόβλημα θα παρουσιασθεί και αν προσπαθήσουμε να τροποποιήσουμε τον κωδικό ενός αθλητή που έχει συμμετάσχει σε κάποιον αγώνα ή τον κωδικό ενός αγώνα στον οποίον έχουν συμμετάσχει κάποιοι αθλητές.

Για να μπορέσουμε να αντιμετωπίσουμε την ανάγκη διαγραφής μιας πλειάδας ή τροποποίησης του πεδίου κλειδιού χωρίς να έχουμε παραβίαση της ακεραιότητας αναφοράς, χρησιμοποιούμε τους όρους *on delete cascade* και *on update cascade* αντίστοιχα, ως εξής :

```
Create table ΣΥΜΜΕΤΟΧΗ
        (Κωδικός_Αθλητή     integer not null,
        Κωδικός_Αγώνα       integer not null,
        Επίδοση              integer,
        primary key(Κωδικός_Αθλητή, Κωδικός_Αγώνα),
```

***foreign key****(Κωδικός_Αθλητή)* ***references*** *ΑΘΛΗΤΗΣ*
***on delete cascade***
***on update cascade****,*
***foreign key****(Κωδικός_Αγώνα)* ***references*** *ΑΓΩΝΑΣ*
***on delete cascade***
***on update cascade****);*

# Οι Όψεις (Views)

Η *όψη (view)* είναι μια σχέση μιας βάσης δεδομένων που δεν έχει δημιουργηθεί με κάποια εντολή create table. Με τις όψεις μπορούμε να εμφανίζουμε ορισμένα μόνο πεδία μιας σχέσης ή και κάποιες άλλες πληροφορίες που αν δεν ήταν οι όψεις θα έπρεπε να δίνουμε κάθε φορά πολύπλοκες εντολές για να δούμε τις πληροφορίες που θέλουμε. Μπορούμε να δημιουργούμε, να τροποποιούμε και να διαγράφουμε όσες όψεις θέλουμε, χωρίς να επηρεάζονται καθόλου τα δεδομένα των σχέσεων στις οποίες αναφέρονται οι όψεις.

Για να δημιουργήσουμε μια όψη, δίνουμε την εντολή *create view as*, ως εξής :

***Create view*** *ΑΘΛΗΤΗΣ_1* ***as***
*(****Select*** *Κωδικός_Αθλητή, Επώνυμο, Ημνία_Γέννησης*
***From*** *ΑΘΛΗΤΗΣ);*

Μετά τον όρο as γράφουμε μέσα σε παρένθεση την εντολή SQL της οποίας το αποτέλεσμα θα δημιουργήσει την όψη. Η παραπάνω όψη εμφανίζει λίγα μόνο από τα πεδία της σχέσης ΑΘΛΗΤΗΣ. Η επόμενη όψη εμφανίζει την μέση τιμή των ατομικών επιδόσεων (ρεκόρ) των αθλητών :

***Create view*** *ΑΘΛΗΤΗΣ_2* ***as***
*(****Select avg****(Ρεκόρ)*
***From*** *ΑΘΛΗΤΗΣ);*

Για να διαγράψουμε μια όψη, δίνουμε την εντολή *drop view*, ως εξής :

***Drop view*** *ΑΘΛΗΤΗΣ_1;*

# Η Εντολή Select

Θα δούμε τώρα τις εντολές της Γλώσσας Χειρισμού Δεδομένων DML (Data Manipulation Language). Η βασικότερη είναι η εντολή *Select*, που χρησιμοποιείται για την εμφάνιση (ανάκληση) δεδομένων από τη βάση δεδομένων. Η γενική σύνταξη της εντολής Select για την αναζήτηση δεδομένων σε μια βάση δεδομένων είναι η εξής :

***Select*** *Πεδίο1, Πεδίο2, ... Πεδίον*
***From*** *Πίνακας1, Πίνακας2, ..., Πίνακαςm*

***Where*** *συνθήκη;*

Τα πεδία που δηλώνουμε στην εντολή Select είναι αυτά που θα εμφανίζονται στο αποτέλεσμα. Είναι αντίστοιχο με την πράξη της προβολής (projection) της σχεσιακής άλγεβρας. Αν θέλουμε να εμφανίζονται όλα τα πεδία θα πρέπει να χρησιμοποιήσουμε τον χαρακτήρα *. Ο όρος *From* δηλώνει τις σχέσεις (πίνακες) στους οποίους θα γίνει η αναζήτηση. Είναι αντίστοιχο με την πράξη του καρτεσιανού γινομένου της σχεσιακής άλγεβρας. Ο όρος *Where* είναι προαιρετικός και περιέχει τη *συνθήκη* που θέλουμε να ικανοποιούν οι εγγραφές (πλειάδες) του αποτελέσματος. Είναι αντίστοιχο με την πράξη της επιλογής (selection) της σχεσιακής άλγεβρας.

Η επόμενη εντολή Select εμφανίζει μόνο τα πεδία Επώνυμο και Όνομα αθλητή καθώς και τον Κωδικό του Συλλόγου στον οποίο ανήκει ο κάθε αθλητής και αυτό γι' όλους ανεξαιρέτως τους αθλητές που υπάρχουν στον πίνακα ΑΘΛΗΤΗΣ :

***Select*** *Επώνυμο, Όνομα, Κωδικός_Συλλόγου*

***From*** *ΑΘΛΗΤΗΣ;*

Επειδή υπάρχει η περίπτωση, ακραία μεν αλλά εφικτή, να υπάρχουν δύο ή και περισσότερες εγγραφές με ίδια τα παραπάνω στοιχεία (πεδία) ενός αθλητή, μπορούμε να χρησιμοποιήσουμε και τον όρο *distinct*, ώστε να μην εμφανίζονται οι όμοιες εγγραφές, ως εξής :

***Select distinct*** *Επώνυμο, Όνομα, Κωδικός_Συλλόγου*

***From*** *ΑΘΛΗΤΗΣ;*

Αν θέλουμε να δούμε όλα τα στοιχεία (πεδία) των αθλητών, θα πρέπει να χρησιμοποιήσουμε το σύμβολο *, ως εξής :

***Select*** ***

***From*** *ΑΘΛΗΤΗΣ;*

Η επόμενη εντολή Select εμφανίζει εκείνους τους αθλητές που έχουν ατομικό ρεκόρ μεγαλύτερο από μια συγκεκριμένη τιμή :

***Select*** *Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Ρεκόρ > 10.05;*

Η επόμενη εντολή Select εμφανίζει με τη χρήση της διάζευξης (λογικό ή – or) εκείνους τους αθλητές που ανήκουν σ' έναν από δύο συλλόγους :

***Select*** *Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Κωδικός_Συλλόγου = 1* ***or*** *Κωδικός_Συλλόγου = 2;*

Η επόμενη εντολή Select εμφανίζει με τη χρήση της σύζευξης (λογικό και – and) εκείνους τους αθλητές που έχουν ατομικό ρεκόρ σε μια περιοχή τιμών :

***Select*** *Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Ρεκόρ > 10.00 and Ρεκόρ < 10.08;*

Θα δούμε τώρα πώς μπορεί να γίνει η ανάκληση δεδομένων από πεδία που περιέχονται σε περισσότερες από μία σχέσεις. Η αναφορά σε πολλές σχέσεις αντιστοιχεί στην πράξη της σύνδεσης (join) της σχεσιακής άλγεβρας.

Για παράδειγμα, για τη συσχέτιση των σχέσεων ΑΘΛΗΤΗΣ και ΣΥΜΜΕΤΟΧΗ χρησιμοποιήσαμε το κοινό πεδίο Κωδικός_Αθλητή, το οποίο είναι πρωτεύον κλειδί στη σχέση ΑΘΛΗΤΗΣ και ξένο κλειδί στη σχέση ΣΥΜΜΕΤΟΧΗ.

Για να βρούμε τώρα ποιοι αθλητές συμμετείχαν σε ποιους αγώνες, δίνουμε την εξής εντολή :

***Select*** *Κωδικός_Αθλητή, Επώνυμο, Όνομα, Κωδικός_Αγώνα*

***From*** *ΑΘΛΗΤΗΣ, ΣΥΜΜΕΤΟΧΗ*

***Where*** *ΑΘΛΗΤΗΣ.Κωδικός_Αθλητή =*

*ΣΥΜΜΕΤΟΧΗ. Κωδικός_Αθλητή;*

Είδαμε στο προηγούμενο παράδειγμα τη χρήση του όρου where με την απαραίτητη συνθήκη για τη δημιουργία της σύνδεσης των δύο σχέσεων (πινάκων). Μπορούμε να ορίσουμε και μια πιο πολύπλοκη συνθήκη για να βρούμε τους αθλητές που συμμετείχαν σε κάποιον συγκεκριμένο αγώνα :

***Select*** *Κωδικός_Αθλητή, Επώνυμο, Όνομα, Κωδικός_Αγώνα*

***From*** *ΑΘΛΗΤΗΣ, ΣΥΜΜΕΤΟΧΗ*

***Where*** *(ΑΘΛΗΤΗΣ.Κωδικός_Αθλητή =*

*ΣΥΜΜΕΤΟΧΗ. Κωδικός_Αθλητή)* ***and*** *Κωδικός_Αγώνα=100;*

# Ο Όρος As

Με τον όρο *as* μπορούμε να μετονομάσουμε πεδία ή σχέσεις αλλά μόνο προσωρινά για τις ανάγκες μιας εντολής Select και όχι μόνιμα. Για παράδειγμα, για να βρούμε τη διαφορά στην επίδοση των αθλητών από το όριο των 10.00, μπορούμε να δώσουμε την εξής εντολή :

***Select*** *Επώνυμο, Όνομα, Κωδικός_Συλλόγου, Ρεκόρ – 10.00* ***as***

*Διαφορά*

***From*** *ΑΘΛΗΤΗΣ;*

Μετονομάσαμε τη διαφορά Ρεκόρ – 10.00 ως Διαφορά, ώστε να μπορούμε να χειριστούμε το αποτέλεσμα της πράξης αυτής με μεγαλύτερη ευκολία αργότερα. Ο όρος as χρησιμοποιείται αναγκαστικά και για την μετονομασία πινάκων, όταν χρειαζόμαστε δύο μεταβλητές για την ίδια σχέση. Για παράδειγμα, για να μπορέσουμε να βρούμε τα στοιχεία (κωδικός, επώνυμο, όνομα) των αθλητών που ανήκουν στον ίδιο σύλλογο με τον αθλητή που έχει κωδικό ίσο με 106, δίνουμε της εξής εντολή :

***Select*** *Κωδικός_Αθλητή, Επώνυμο, Όνομα*

***From** ΑΘΛΗΤΗΣ **as** Α, ΑΘΛΗΤΗΣ **as** Β*

***Where** Α.Κωδικός_Συλλόγου = Β.Κωδικός_Συλλόγου **and***

*Β.Κωδικός_Αθλητή = 106;*

# Ο Όρος Order By

Με τον όρο *order by* μπορούμε να ταξινομήσουμε το αποτέλεσμα μιας εντολής Select ως προς κάποιο πεδίο. Για παράδειγμα, για να εμφανίσουμε μια λίστα των αθλητών ταξινομημένη ως προς επώνυμο πρώτα και μετά ως προς όνομα, δίνουμε την εξής εντολή :

***Select** Επώνυμο, Όνομα, Κωδικός_Συλλόγου*

***From** ΑΘΛΗΤΗΣ*

***Order by** Επώνυμο, Όνομα;*

Η παραπάνω εντολή θα ταξινομήσει πρώτα ως προς το πεδίο Επώνυμο και όπου υπάρχουν ίδια Επώνυμα θα ταξινομήσει ως προς το πεδίο Όνομα.

Εξ ορισμού η ταξινόμηση γίνεται κατ' αύξουσα σειρά. Μπορούμε να χρησιμοποιήσουμε και τους όρους *asc* για αύξουσα και *desc* για φθίνουσα ταξινόμηση, ως εξής :

***Select** Επώνυμο, Όνομα, Κωδικός_Συλλόγου*

***From** ΑΘΛΗΤΗΣ*

***Order by** Κωδικός_Συλλόγου **desc**, Επώνυμο **asc**;*

Η παραπάνω εντολή θα ταξινομήσει πρώτα ως προς το πεδίο Κωδικός_Συλλόγου κατά φθίνουσα σειρά και όπου υπάρχουν ίδιοι κωδικοί συλλόγου θα ταξινομήσει ως προς το πεδίο Επώνυμο κατ' αύξουσα σειρά.

# Οι Συναρτήσεις Ομαδοποίησης

Η SQL χρησιμοποιεί μερικές πολύ χρήσιμες συναρτήσεις, που ονομάζονται *συναρτήσεις ομαδοποίησης (aggregate functions)* και οι οποίες δέχονται ένα σύνολο τιμών και επιστρέφουν μία τιμή. Οι συναρτήσεις αυτές είναι οι εξής :

| **Συνάρτηση Ομαδοποίησης** | **Αντίστοιχος Όρος στην SQL** |
|---|---|
| Απαρίθμηση | Count |
| Άθροισμα | Sum |
| Μέσος Όρος | Avg |
| Μέγιστη Τιμή | Max |
| Ελάχιστη Τιμή | Min |

Οι συναρτήσεις Sum και Avg εργάζονται μόνο με αριθμητικές τιμές, ενώ οι υπόλοιπες συναρτήσεις μπορούν να δεχθούν και αλφαριθμητικές τιμές. Για να βρούμε τον συνολικό αριθμό των αθλητών, δίνουμε την εξής εντολή :

***Select count**(*)*

***From** ΑΘΛΗΤΗΣ;*

Το αποτέλεσμα θα είναι ένας μόνο αριθμός, δηλ. ο συνολικός αριθμός των αθλητών (εγγραφών) της σχέσης ΑΘΛΗΤΗΣ. Για να βρούμε τον αθλητή που έχει την καλύτερη επίδοση, δίνουμε την εξής εντολή :

***Select** Κωδικός_Αθλητή, Επώνυμο, Όνομα, **max**(Ρεκόρ)*

***From** ΑΘΛΗΤΗΣ;*

Εδώ η συνάρτηση max() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου Ρεκόρ και επιστρέφει την μέγιστη τιμή μαζί με τα στοιχεία του αθλητή που έχει το καλύτερο ρεκόρ. Για να βρούμε τον αθλητή που έχει την χαμηλότερη επίδοση, δίνουμε την εξής εντολή :

***Select** Κωδικός_Αθλητή, Επώνυμο, Όνομα, **min**(Ρεκόρ)*

***From** ΑΘΛΗΤΗΣ;*

Εδώ η συνάρτηση min() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου Ρεκόρ και επιστρέφει την ελάχιστη τιμή μαζί με τα στοιχεία του αθλητή που έχει το χειρότερο ρεκόρ. Για να βρούμε τον μέσο όρο των επιδόσεων των αθλητών, δίνουμε την εξής εντολή :

***Select avg**(Ρεκόρ)*

***From** ΑΘΛΗΤΗΣ;*

Εδώ η συνάρτηση avg() δέχεται σαν είσοδο όλες τις αριθμητικές τιμές του πεδίου Ρεκόρ και επιστρέφει την μέση τιμή των επιδόσεων όλων των αθλητών. Και εδώ το αποτέλεσμα θα είναι ένας μόνο αριθμός.

# Ο Όρος Group By

Με τον όρο *group by* μπορούμε να ξεχωρίσουμε (απομονώσουμε) τις εγγραφές μιας σχέσης σε ανεξάρτητα υποσύνολα και μετά να εφαρμόσουμε μια συνάρτηση ομαδοποίησης σε κάθε υποσύνολο. Για παράδειγμα, σ' έναν πίνακα ΠΕΛΑΤΩΝ μπορούμε να κάνουμε ομαδοποίηση σύμφωνα με το πεδίο Πόλη και έτσι να βρούμε πόσοι πελάτες υπάρχουν σε κάθε πόλη ή ποιος είναι ο μέσος όρος του υπολοίπου από τους πελάτες της κάθε πόλης ή ποιο είναι το μέγιστο υπόλοιπο από τους πελάτες της κάθε πόλης κοκ.

Μπορούμε ακόμα να εφαρμόσουμε και συνθήκες στην ομαδοποίηση, όπως αν θέλουμε να βρούμε σε ποιες πόλεις ο μέσος όρος του υπολοίπου των πελατών είναι μεγαλύτερος από 1.000 € κοκ. Παρατηρούμε ότι η ομαδοποίηση μπορεί να εφαρμοσθεί μόνο σε πεδία τα οποία έχουν επαναλαμβανόμενες τιμές σ' έναν πίνακα. Αυτό σημαίνει ότι η ομαδοποίηση έχει νόημα να γίνει με το πεδίο πόλη

αλλά προφανώς δεν έχει νόημα να γίνει με το πεδίο επώνυμο, γιατί τα επώνυμα των πελατών θα είναι διαφορετικά μεταξύ τους.

Όταν χρησιμοποιούμε τον όρο group by για να δηλώσουμε ότι θέλουμε να κάνουμε ομαδοποίηση σύμφωνα με κάποιο πεδίο, τότε πρώτα γίνεται αυτόματα αύξουσα ταξινόμηση σύμφωνα με το πεδίο αυτό, δημιουργούνται τα αναγκαία υποσύνολα και μετά εφαρμόζονται στα υποσύνολα αυτά οι συναρτήσεις ομαδοποίησης που έχουμε ορίσει.

Θεωρούμε συνεπώς τον εξής πίνακα ΠΕΛΑΤΗΣ :

```
Create table ΠΕΛΑΤΗΣ
        (Κωδικός_Πελάτη     integer not null,
        Επώνυμο             char(20) not null,
        Όνομα               char(15),
        Πόλη                char(15),
        Ημνία_Γέννησης      date,
        Υπόλοιπο            numeric(4, 2),
        primary key (Κωδικός_Πελάτη));
```

Για να βρούμε τον μέσο όρο του υπολοίπου των πελατών ανά πόλη, δίνουμε την εξής εντολή :

```
Select Πόλη, avg(Υπόλοιπο)
From ΠΕΛΑΤΗΣ
Group by Πόλη;
```

Η παραπάνω εντολή χωρίζει τη σχέση ΠΕΛΑΤΗΣ σε τόσα υποσύνολα όσες είναι οι υπάρχουσες πόλεις του πίνακα ΠΕΛΑΤΗΣ, όπου το κάθε υποσύνολο αναφέρεται σε μια συγκεκριμένη πόλη. Στη συνέχεια, εφαρμόζεται η συνάρτηση ομαδοποίησης avg() (μέσος όρος) σε κάθε υποσύνολο και το αποτέλεσμα θα είναι μια σχέση με τόσες εγγραφές όσες είναι και οι υπάρχουσες πόλεις του πίνακα ΠΕΛΑΤΗΣ.

Κάθε εγγραφή του αποτελέσματος θα περιέχει δύο πεδία, όπου το ένα θα είναι η πόλη και το άλλο ο μέσος όρος του υπολοίπου των πελατών για την κάθε πόλη. Για να βρούμε τις πόλεις στις οποίες ο μέσος όρος του υπολοίπου των πελατών ανά πόλη είναι μεγαλύτερος από 1.000 €, δίνουμε την εξής εντολή :

```
Select Πόλη, avg(Υπόλοιπο)
From ΠΕΛΑΤΗΣ
Group by Πόλη
Having avg(Υπόλοιπο) > 1000;
```

Ο όρος *having* χρησιμοποιείται στις συνθήκες που εφαρμόζονται σε κάθε υποσύνολο που δημιουργείται από τον όρο group by και όχι σε κάθε εγγραφή, όπως συμβαίνει με τη συνθήκη του όρου where. Η παρακάτω εντολή εμφανίζει

το σύνολο των πελατών που βρίσκονται στην πόλη της Λάρισας και στην πόλη της Κοζάνης, ξεχωριστά για κάθε πόλη, ως εξής :

```
Select Πόλη, count(*)
 From ΠΕΛΑΤΗΣ
 Group by Πόλη
Having Πόλη='Λάρισα' or Πόλη='Κοζάνη';
```

# Ο Όρος Intersect (Τομή)

Επειδή οι σχέσεις αντιστοιχούν σε σύνολα, μπορούμε να χρησιμοποιήσουμε τις πράξεις μεταξύ συνόλων και στις σχέσεις. Η πράξη της *τομής* στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο *Intersect*. Για να μπορέσουμε να εφαρμόσουμε την πράξη της τομής, όπως και τις πράξεις της ένωσης αλλά και της διαφοράς, ανάμεσα σε δύο σχέσεις, θα πρέπει αυτές να είναι *συμβατές*, δηλ. να έχουν τα ίδια πεδία ή πεδία με το ίδιο πεδίο ορισμού.

Για παράδειγμα, για να βρούμε τους κωδικούς των αθλητών που δεν ανήκουν σε κάποιο σύλλογο και που έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
(Select Κωδικός_Αθλητή
 From ΑΘΛΗΤΗΣ
 Where Κωδικός_Συλλόγου <> 202)
Intersect
(Select Κωδικός_Αθλητή
 From ΣΥΜΜΕΤΟΧΗ
 Where Κωδικός_Αγώνα = 301);
```

Η πράξη της τομής στο παραπάνω παράδειγμα είναι εφικτή, εφόσον οι δύο αυτές σχέσεις έχουν τα ίδια πεδία και δημιουργεί μια τρίτη σχέση που περιέχει τις κοινές πλειάδες των δύο αυτών σχέσεων.

# Ο Όρος Union (Ένωση)

Η πράξη της *ένωσης* στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο *Union*. Για παράδειγμα, για να βρούμε τους κωδικούς των αθλητών που έχουν ρεκόρ μεγαλύτερο από κάποια συγκεκριμένη τιμή ή που έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
(Select Κωδικός_Αθλητή
 From ΑΘΛΗΤΗΣ
 Where Ρεκόρ > 10.40)
```

```
Union
(Select Κωδικός_Αθλητή
 From ΣΥΜΜΕΤΟΧΗ
 Where Κωδικός_Αγώνα = 302);
```

# Ο Όρος Except (Διαφορά)

Η πράξη της *διαφοράς* στη σχεσιακή άλγεβρα επιτυγχάνεται στην SQL με τον όρο *Except*. Για παράδειγμα, για να βρούμε τους κωδικούς των αθλητών που δεν έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
(Select Κωδικός_Αθλητή
 From ΑΘΛΗΤΗΣ)
Except
(Select Κωδικός_Αθλητή
 From ΣΥΜΜΕΤΟΧΗ
 Where Κωδικός_Αγώνα = 302);
```

# Οι Όροι In και Not In

Οι εντολές της SQL έχουν την ιδιότητα να εφαρμόζονται σε σχέσεις και το αποτέλεσμά τους να αποτελεί κι αυτό μια σχέση (ιδιότητα της κλειστότητας). Έτσι, μπορούμε να εμφωλιάσουμε εντολές ώστε το αποτέλεσμα (έξοδος) της μιας εντολής να αποτελεί είσοδο για μια άλλη εντολή.

Με τον όρο *in* μπορούμε να ελέγξουμε αν μια πλειάδα ανήκει σε μια σχέση η οποία δημιουργείται από μια φωλιασμένη εντολή. Ο όρος in είναι κάτι αντίστοιχο με το γνωστό σύμβολο των μαθηματικών ανήκει σε ($\in$). Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που έχουν συμμετάσχει σε κάποιον συγκεκριμένο αγώνα, δίνουμε την εξής εντολή :

```
Select Επώνυμο, Όνομα
From ΑΘΛΗΤΗΣ
Where Κωδικός_Αθλητή in (Select Κωδικός_Αθλητή
                         From ΣΥΜΜΕΤΟΧΗ
                         Where Κωδικός_Αγώνα = 302);
```

Θα εκτελεσθεί πρώτα η εντολή που βρίσκεται μέσα στις παρενθέσεις και από τη σχέση που θα δημιουργηθεί, η οποία θα περιέχει μια λίστα με τους κωδικούς των αθλητών που συμμετείχαν στον συγκεκριμένο αγώνα (ΣΥΜΜΕΤΟΧΗ), θα γίνει σύγκριση με τους κωδικούς των αθλητών που υπάρχουν στη σχέση ΑΘΛΗΤΗΣ, ώστε να εμφανισθούν τα Επώνυμα και τα Ονόματα των αθλητών που συμμετείχαν στον συγκεκριμένο αγώνα.

Υπάρχει και ο όρος *not in* για να ελέγξουμε αν μια πλειάδα δεν ανήκει σε μια σχέση η οποία δημιουργείται από μια φωλιασμένη εντολή. Ο όρος not in είναι κάτι αντίστοιχο με το γνωστό σύμβολο των μαθηματικών δεν ανήκει σε (∉). Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που δεν ανήκουν σε κάποιους συγκεκριμένους συλλόγους, δίνουμε την εξής εντολή :

***Select*** *Επώνυμο, Όνομα*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Κωδικός_Συλλόγου* ***not in*** *(1, 2);*

# Οι Όροι Some και All

Χρησιμοποιούμε τον όρο *some* για να απαντήσουμε σε ερωτήματα όπου η τιμή ενός πεδίου θέλουμε να συγκριθεί με την τιμή ενός πεδίου *μίας τουλάχιστον* πλειάδας μιας σχέσης που προκύπτει από μια φωλιασμένη εντολή. Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που έχουν ατομικό ρεκόρ μικρότερο από έναν τουλάχιστον αθλητή ενός συγκεκριμένου συλλόγου, δίνουμε την εξής εντολή :

***Select*** *Επώνυμο, Όνομα*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Ρεκόρ* < ***some (Select*** *Ρεκόρ*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Κωδικός_Συλλόγου = 3);*

Χρησιμοποιούμε τον όρο *all* για να απαντήσουμε σε ερωτήματα όπου η τιμή ενός πεδίου θέλουμε να συγκριθεί με την τιμή ενός πεδίου *όλων των* πλειάδων μιας σχέσης που προκύπτει από μια φωλιασμένη εντολή. Για παράδειγμα, για να βρούμε τα ονόματα των αθλητών που έχουν ατομικό ρεκόρ μικρότερο απ' όλους τους αθλητές ενός συγκεκριμένου συλλόγου, δίνουμε την εξής εντολή :

***Select*** *Επώνυμο, Όνομα*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Ρεκόρ* < ***all (Select*** *Ρεκόρ*

***From*** *ΑΘΛΗΤΗΣ*

***Where*** *Κωδικός_Συλλόγου = 3);*

# Η Εντολή Insert Into

Οι εντολές της SQL που έχουμε συναντήσει μέχρι τώρα είχαν να κάνουν με την αναζήτηση και την εμφάνιση στοιχείων από μια βάση δεδομένων. Υπάρχουν, όμως, και εντολές για να εισάγουμε, διαγράψουμε και τροποποιήσουμε δεδομένα από μια βάση. Με την εντολή *insert into* μπορούμε να καταχωρήσουμε πλειάδες σε μια ήδη υπάρχουσα σχέση. Ακολουθεί ένα παράδειγμα :

***Insert into*** *ΑΘΛΗΤΗΣ (Κωδικός_Αθλητή, Επώνυμο, Όνομα, Ρεκόρ, Ημνία_Γέννησης, Κωδικός_Συλλόγου)*

***values*** *(126, 'Νταμπίζας', 'Νικόλαος', 10.57, null, 306);*

Παρατηρούμε ότι πρέπει να τηρήσουμε με προσοχή τη σειρά των πεδίων και τις αντίστοιχες τιμές τους (αλφαριθμητικές, αριθμητικές, ημερομηνίας κοκ). Για τα πεδία που δεν έχουμε κάποια τιμή, μπορούμε να δηλώσουμε τιμή null, εφόσον φυσικά αυτό επιτρέπεται.

# Η Εντολή Delete From

Με την εντολή *delete from* μπορούμε να διαγράψουμε ολόκληρες πλειάδες και όχι μεμονωμένα πεδία (στήλες). Για παράδειγμα, για να διαγράψουμε όλους τους αθλητές που έχουν ατομικό ρεκόρ μεγαλύτερο από κάποια συγκεκριμένη τιμή, δίνουμε την εξής εντολή :

***Delete from*** *ΑΘΛΗΤΗΣ*

***Where*** *Ρεκόρ > 11.00;*

Για να διαγράψουμε όλες τις πλειάδες μιας σχέσης, αλλά όχι και την ίδια την σχέση, δίνουμε την εξής εντολή :

***Delete from*** *ΑΘΛΗΤΗΣ;*

# Η Εντολή Update

Με την εντολή *update* μπορούμε να τροποποιήσουμε την τιμή κάποιων πεδίων από ορισμένες ή και απ' όλες τις πλειάδες μιας σχέσης. Για παράδειγμα, για να αλλάξουμε τον κωδικό συλλόγου σε 320 για όσους αθλητές ανήκουν στον σύλλογο που έχει κωδικό 307, δίνουμε την εξής εντολή :

***Update*** *ΑΘΛΗΤΗΣ*

***Set*** *Κωδικός_Συλλόγου = 320*

***Where*** *Κωδικός_Συλλόγου = 307;*

Επίσης, για να μηδενίσουμε τα ρεκόρ όλων των αθλητών, δίνουμε την εξής εντολή : ***Update*** *ΑΘΛΗΤΗΣ*

***Set*** *Ρεκόρ = 00.00;*