

## Η Δομημένη Γλώσσα Ερωτημάτων SQL

### Τι Είναι η SQL

Η SQL αποτελεί μια στάνταρτ γλώσσα του ANSI για να μπορούμε να έχουμε πρόσβαση σε βάσεις δεδομένων.

- Τα αρχικά SQL σημαίνουν **Structured Query Language**, δηλ. *Δομημένη Γλώσσα Ερωτημάτων*.
- Η SQL μάς δίνει τη δυνατότητα να έχουμε πρόσβαση σε μια βάση δεδομένων (database).
- Η SQL αποτελεί μια στάνταρτ γλώσσα του ANSI (ANSI standard language).
- Η SQL μπορεί να εκτελέσει ερωτήματα (queries) σχετικά με μια βάση δεδομένων.
- Η SQL μπορεί να ανακτήσει δεδομένα από μια βάση δεδομένων.
- Η SQL μπορεί να εισαγάγει νέες εγγραφές σε μια βάση δεδομένων.
- Η SQL μπορεί να διαγράψει εγγραφές από μια βάση δεδομένων.
- Η SQL μπορεί να ενημερώσει εγγραφές σε μια βάση δεδομένων.
- Η SQL είναι πολύ εύκολη στην εκμάθηση.

Η SQL αποτελεί ένα στάνταρτ του ANSI (American National Standards Institute) για να μπορούμε να έχουμε πρόσβαση σε συστήματα βάσεων δεδομένων. Οι εντολές της SQL χρησιμοποιούνται για να ανακτήσουμε (retrieve) και να ενημερώσουμε (update) δεδομένα σε μια βάση δεδομένων (database).

Η SQL συνεργάζεται με προγράμματα βάσεων δεδομένων όπως είναι τα εξής : Access, Informix, Microsoft SQL Server, Oracle, Sybase και πολλά άλλα.

### Οι Πίνακες Βάσεων Δεδομένων (Database Tables)

Οι βάσεις δεδομένων (databases) περιέχουν αντικείμενα (objects) που ονομάζονται Πίνακες (Tables). Οι Εγγραφές (Records) των δεδομένων απο-θηκεύονται σ' αυτούς τους πίνακες. Οι Πίνακες αναγνωρίζονται με τα ονόματά τους, όπως "Persons", "Orders", "Suppliers" κ.ά.

Οι Πίνακες περιέχουν Στήλες (Columns) και Γραμμές (Rows) με δε-δομένα. Οι Γραμμές (Rows) περιέχουν εγγραφές (records), όπως μία εγγραφή για κάθε άτομο. Οι Στήλες (Columns) περιέχουν δεδομένα, όπως First Name, Last Name, Address και City.

Ακολουθεί ένα παράδειγμα ενός Πίνακα που ονομάζεται "Persons" :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Τα LastName, FirstName, Address και City είναι οι Στήλες (Columns) του πίνακα. Οι Γραμμές (Rows) περιέχουν τρεις εγγραφές για τρία άτομα.

## Τα Ερωτήματα της SQL (SQL Queries)

Με την SQL, μπορούμε να κάνουμε ένα ερώτημα (Query) σε μια βάση δεδομένων και να έχουμε ένα αποτέλεσμα (Result) σε μορφή πίνακα (tabular form).

Ένα ερώτημα σαν το εξής :

*SELECT LastName FROM Persons*

θα δώσει ένα αποτέλεσμα σαν το εξής :

LastName
Παπαδόπουλος
Αντωνιάδης
Γεωργιάδης

Πρέπει να έχουμε υπόψη μας ότι μερικά συστήματα βάσεων δεδομένων απαιτούν το σύμβολο ; (semicolon) στο τέλος μιας εντολής SQL. Δεν θα χρησιμοποιήσουμε εδώ το σύμβολο ;.

## Χειρισμός Δεδομένων της SQL (Data Manipulation)

Όπως υπονοεί και το όνομά της, η SQL είναι μια σύνταξη για την εκτέλεση ερωτημάτων (queries). Αλλά η γλώσσα της SQL περιλαμβάνει επίσης μια σύνταξη για την ενημέρωση εγγραφών, την εισαγωγή νέων εγγραφών και τη διαγραφή υπαρχόντων εγγραφών.

Αυτές οι εντολές ερωτημάτων και ενημέρωσης αποτελούν μαζί τη *Γλώσσα Χειρισμού Δεδομένων (Data Manipulation Language, DML)* που αποτελεί κομμάτι της SQL :

- **SELECT** - εξάγει δεδομένα από μια βάση δεδομένων.
- **UPDATE** - ενημερώνει δεδομένα σε μια βάση δεδομένων.
- **DELETE** - διαγράφει δεδομένα από μια βάση δεδομένων.
- **INSERT** - εισάγει νέα δεδομένα σε μια βάση δεδομένων.

## Ορισμός Δεδομένων της SQL (Data Definition)

Η *Γλώσσα Ορισμού Δεδομένων (Data Definition Language, DDL)*, που αποτελεί μέρος της SQL, επιτρέπει τη δημιουργία και τη διαγραφή πινάκων μιας βάσης δεδομένων. Μπορούμε επίσης να ορίσουμε indexes (keys), να καθορίσουμε συνδέσμους (links) ανάμεσα στους πίνακες και να επιβάλλουμε περιορισμούς ανάμεσα στους πίνακες μιας βάσης δεδομένων.

Οι σημαντικότερες εντολές DDL στην SQL είναι οι εξής :

- **CREATE TABLE** - δημιουργεί έναν νέον πίνακα σε μια βάση δεδομένων.
- **ALTER TABLE** - τροποποιεί έναν πίνακα σε μια βάση δεδομένων.
- **DROP TABLE** - διαγράφει έναν πίνακα από μια βάση δεδομένων.
- **CREATE INDEX** - δημιουργεί έναν index (search key).
- **DROP INDEX** - διαγράφει έναν index.

## Η SQL και οι Ενεργές Σελίδες Διακομιστή

Η SQL αποτελεί ένα σημαντικό κομμάτι της ASP, επειδή το *Ενεργό Αντικείμενο Δεδομένων* (*Active Data Object, ADO*) που χρησιμοποιείται στην *ASP* (*Active Server Pages*) για να μπορούμε να έχουμε πρόσβαση σε βάσεις δεδομένων, βασίζεται στην SQL για την πρόσβαση στα δεδομένα.

### Δημιουργία Βάσης Δεδομένων και Πίνακα

Για να δημιουργήσουμε μια βάση δεδομένων, μπορούμε να δώσουμε την εξής εντολή :

**CREATE DATABASE** όνομα\_βάσης\_δεδομένων

Για να δημιουργήσουμε έναν πίνακα σε μια βάση δεδομένων, μπορούμε να δώσουμε την εξής εντολή :

**CREATE TABLE** όνομα\_πίνακα

(  
    όνομα\_στήλης\_1 τύπος\_δεδομένων,  
    όνομα\_στήλης\_2 τύπος\_δεδομένων,  
    ...  
)

### Παράδειγμα

Θα δημιουργήσουμε έναν πίνακα με όνομα "Person", με τέσσερις στήλες με ονόματα "LastName", "FirstName", "Address" και "Age" :

**CREATE TABLE** Person

(  
    LastName varchar,  
    FirstName varchar,  
    Address varchar,  
    Age number  
)

Θα ορίσουμε ένα μέγιστο μήκος για μερικές στήλες :

**CREATE TABLE** Person

(

*LastName varchar(30),*

*FirstName varchar,*

*Address varchar,*

*Age number(3)*

)

Ο τύπος δεδομένων (data type) καθορίζει τι είδος δεδομένων θα περιέχει η στήλη.

### **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.

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* ('Τρεβενών', '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);*

### Διαγραφή Βάσης Δεδομένων και Πίνακα

Για να διαγράψουμε μια βάση δεδομένων, χρησιμοποιούμε την εξής εντολή :

***DROP DATABASE*** όνομα\_βάσης\_δεδομένων

Για να διαγράψουμε έναν πίνακα, χρησιμοποιούμε την εξής εντολή :

***DROP TABLE*** όνομα\_πίνακα

Για να διαγράψουμε τα δεδομένα ενός πίνακα χωρίς να διαγράψουμε τον πίνακα, χρησιμοποιούμε την εξής εντολή :

***DELETE TABLE*** όνομα\_πίνακα

### Η Εντολή Alter Table

Η εντολή ALTER TABLE χρησιμοποιείται για να προσθέσουμε ή να διαγράψουμε στήλες από έναν υπάρχοντα πίνακα.

Η σύνταξή της για τις δύο αυτές περιπτώσεις είναι ως εξής :

***ALTER TABLE*** όνομα\_πίνακα ***ADD*** όνομα\_στήλης τύπος\_δεδομένων

***ALTER TABLE*** όνομα\_πίνακα ***DROP*** όνομα\_στήλης

Πίνακας Person :

LastName	FirstName	Address
Σταύρου	Μαρίνα	Σαρανταπόρου 10

### Παράδειγμα

Για να προσθέσουμε μια στήλη με όνομα "City" στον πίνακα "Person", δίνουμε την εξής εντολή :

*ALTER TABLE Person ADD City varchar(30)*

Αποτέλεσμα :

LastName	FirstName	Address	City
Σταύρου	Μαρίνα	Σαρανταπόρου 10	

### Παράδειγμα

Για να διαγράψουμε (drop) τη στήλη "Address" του πίνακα "Person", δίνουμε την εξής εντολή :

*ALTER TABLE Person DROP Address*

Αποτέλεσμα :

LastName	FirstName	City
Σταύρου	Μαρίνα	

## Η Εντολή Select της SQL

Η εντολή SELECT επιλέγει στήλες (columns) δεδομένων από μια βάση δεδομένων. Το αποτέλεσμα αποθηκεύεται σε μορφή πίνακα και αποκαλείται result set. Την χρησιμοποιούμε για να εμφανίζουμε (επιλέγουμε) πληροφορίες από έναν πίνακα ως εξής :

***SELECT*** ονόματα\_στηλών ***FROM*** όνομα\_πίνακα

### Παράδειγμα : Επιλογή Στηλών από έναν Πίνακα

Για να επιλέξουμε τις στήλες "LastName" και "FirstName", χρησιμοποιούμε μια εντολή SELECT, ως εξής :

*SELECT LastName, FirstName FROM Persons*



Το αποτέλεσμα :

LastName	FirstName
Παπαδόπουλος	Δημήτριος
Αντωνιάδης	Αντώνιος
Γεωργιάδης	Νικόλαος

Παράδειγμα : Επιλογή όλων των Στηλών

Για να επιλέξουμε όλες τις στήλες από τον πίνακα "Person", χρησιμοποιούμε το σύμβολο \* αντί για όνομα στήλης, ως εξής :

*SELECT \* FROM Persons*

Το αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

## To Where Clause της SQL

Το WHERE clause χρησιμοποιείται για να καθορίσουμε ένα κριτήριο επιλογής (selection criteria). Για να μπορέσουμε να επιλέξουμε δεδομένα υπό συνθήκη από έναν πίνακα, πρέπει να προσθέσουμε ένα WHERE clause σε μια εντολή SELECT, ως εξής :

*SELECT* στήλη *FROM* πίνακα *WHERE* στήλη συνθήκη τιμή

Με το WHERE clause, μπορούμε να χρησιμοποιήσουμε αυτές τις συνθήκες :

Τελεστής (Operator)	Συνθήκη (Condition)
=	Ίσο
<>	Όχι ίσο
>	Μεγαλύτερο από
<	Μικρότερο από

>=	Μεγαλύτερο από ή ίσο με
<=	Μικρότερο από ή ίσο με
LIKE	Επεξηγείται παρακάτω

Σε μερικές εκδόσεις της SQL, ο τελεστής για το όχι ίσο (<>), μπορεί να γραφεί ως εξής : !=.

#### Παράδειγμα : Επιλογή Ατόμων από μια Πόλη

Για να επιλέξουμε μόνο τα άτομα που κατοικούν στην πόλη Φλώρινα, προσθέτουμε ένα WHERE clause στην εντολή SELECT, ως εξής :

*SELECT \* FROM Persons WHERE City='Φλώρινα'*

Το αποτέλεσμα :

LastName	FirstName	Address	City	Year
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα	1951
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα	1978

Εχουμε χρησιμοποιήσει μονά εισαγωγικά (single quotes) στις τιμές των συνθηκών στα παραδείγματα. Η SQL χρησιμοποιεί μονά εισαγωγικά στις αλφαριθμητικές τιμές (κείμενο). Τα περισσότερα συστήματα βάσεων δεδομένων αποδέχονται και τα διπλά εισαγωγικά (double quotes). Οι αριθμητικές τιμές δεν πρέπει να περικλείονται σε εισαγωγικά.

#### Για τις τιμές κειμένου (αλφαριθμητικές) :

Αυτό είναι σωστό :

*SELECT \* FROM Persons WHERE FirstName='Δημήτριος'*

Αυτό δεν είναι σωστό :

*SELECT \* FROM Persons WHERE FirstName=Δημήτριος*

#### Για τις αριθμητικές τιμές :

Αυτό είναι σωστό :

*SELECT \* FROM Persons WHERE Year>1965*

Αυτό δεν είναι σωστό :

*SELECT \* FROM Persons WHERE Year>'1965'*

## Η Συνθήκη LIKE

Η συνθήκη LIKE χρησιμοποιείται για να καθορίσουμε μια αναζήτηση για ένα υπόδειγμα (pattern) σε μια στήλη.

Η σύνταξη είναι ως εξής :

**SELECT** στήλη **FROM** πίνακα **WHERE** στήλη **LIKE** υπόδειγμα

Μπορούμε να χρησιμοποιήσουμε το σύμβολο "%" για να ορίσουμε χαρακτήρες μπαλαντέρ (wildcards) πριν και μετά από το υπόδειγμα.

Παράδειγμα : Επιλογή Ατόμων με Υπόδειγμα Ονόματος

Η επόμενη εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους αρχίζει από 'Ο'.

**SELECT \* FROM Persons WHERE FirstName LIKE 'O%'**

Η επόμενη εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους τελειώνει σε 'α'.

**SELECT \* FROM Persons WHERE FirstName LIKE '%α'**

Η επόμενη εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους περιέχει το 'γα'.

**SELECT \* FROM Persons WHERE FirstName LIKE '%γα%'**

Όλα τα παραπάνω παραδείγματα θα επιστρέψουν το εξής αποτέλεσμα :

LastName	FirstName	Address	City	Year
Δημητρίου	Ολγα	Ταγμ. Ναούμ 20	Φλώρινα	1951

## Οι Λογικοί Τελεστές And και Or

Τα AND και OR ενώνουν δύο ή περισσότερες συνθήκες σ' ένα WHE-RE clause. Ο τελεστής AND εμφανίζει μια γραμμή αν ΟΛΕΣ οι συνθήκες είναι αληθείς (true), ενώ ο τελεστής OR εμφανίζει μια γραμμή αν ΟΠΟΙΑΔΗΠΟΤΕ από τις συνθήκες είναι αληθής (true).

Ο αρχικός πίνακας είναι ο εξής :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα

Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα
------------	----------	---------------	---------

#### Παράδειγμα

Χρησιμοποιούμε το AND για να εμφανίσουμε αυτούς που το όνομά τους είναι "Νικόλαος" και το επώνυμό τους είναι "Γεωργιάδης" :

*SELECT \* FROM Persons*

*WHERE FirstName='Νικόλαος'*

*AND LastName='Γεωργιάδης'*

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

#### Παράδειγμα

Χρησιμοποιούμε το OR για να εμφανίσουμε αυτούς που το όνομά τους είναι "Νικόλαος" ή το επώνυμό τους είναι "Γεωργιάδης" :

*SELECT \* FROM Persons*

*WHERE firstname='Νικόλαος'*

*OR lastname='Γεωργιάδης'*

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

#### Παράδειγμα

Μπορούμε επίσης να συνδυάσουμε τα AND και OR, χρησιμοποιώντας παρενθέσεις για να σχηματίσουμε πολύπλοκες εκφράσεις :

*SELECT \* FROM Persons WHERE*

*(FirstName='Αντώνιος' OR FirstName='Νικόλαος')*

*AND LastName='Γεωργιάδης'*

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

### Ο Τελεστής Between ... And

Ο τελεστής BETWEEN ... AND επιλέγει μια περιοχή δεδομένων ανάμεσα σε δύο τιμές. Οι τιμές μπορεί να είναι αριθμοί, κείμενο ή ημερομηνίες.

**SELECT** όνομα\_στήλης **FROM** όνομα\_πίνακα

**WHERE** όνομα\_στήλης

**BETWEEN** τιμή1 **AND** τιμή2

Αρχικός πίνακας :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα
Μαρκόπουλος	Νικόλαος	Φον Κάραγιαν 20	Κοζάνη

### Παράδειγμα

Για να εμφανίσουμε τα άτομα που βρίσκονται αλφαβητικά ανάμεσα στους "Γεωργιάδης" και "Μαρκόπουλος", αλλά και να τους περιλαμβάνουν :

**SELECT \* FROM Persons WHERE LastName**

**BETWEEN 'Γεωργιάδης' AND 'Μαρκόπουλος'**

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα
Μαρκόπουλος	Νικόλαος	Φον Κάραγιαν 20	Κοζάνη

### Παράδειγμα

Για να εμφανίσουμε τα άτομα που βρίσκονται έξω από την περιοχή που χρησιμοποιήσαμε στο προηγούμενο παράδειγμα, χρησιμοποιούμε τον τελεστή NOT :

*SELECT \* FROM Persons WHERE LastName*

*NOT BETWEEN 'Γεωργιάδης' AND 'Μαρκόπουλος'*

Αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα

### Η Λέξη Κλειδί Distinct

Η λέξη κλειδί DISTINCT χρησιμοποιείται για να επιστρέφει μόνο διακριτές (διαφορετικές) (distinct, different) τιμές. Η εντολή SELECT της SQL επιστρέφει στοιχεία από τις στήλες ενός πίνακα, αλλά τι μπορούμε να κάνουμε αν θέλουμε να επιλέξουμε μόνο διακριτά στοιχεία (distinct elements);

Στην SQL, αυτό που πρέπει να κάνουμε είναι να προσθέσουμε μια λέξη κλειδί DISTINCT στην εντολή SELECT, ως εξής :

*SELECT DISTINCT ονόματα\_στηλών FROM όνομα\_πίνακα*

### Παράδειγμα

Επιλογή εταιρειών από έναν πίνακα παραγγελιών.

Ενας απλός πίνακας παραγγελιών :

Company	OrderNumber
Line Computers	3412
Sony	2312
Algorithm	4678
Sony	6798

Η επόμενη εντολή SQL :

*SELECT Company FROM Orders*

θα δώσει αυτό το αποτέλεσμα :

Company
Line Computers
Sony
Algorithm
Sony

Βλέπουμε ότι η εταιρεία Sony εμφανίζεται δύο φορές στο αποτέλεσμα. Μερικές φορές δεν το θέλουμε αυτό.

### Παράδειγμα

Επιλογή ξεχωριστών εταιρειών από έναν πίνακα παραγγελιών.

Η επόμενη εντολή SQL :

*SELECT DISTINCT Company FROM Orders*

θα δώσει αυτό το αποτέλεσμα :

Company
Line Computers
Sony
Algorithm

Τώρα η εταιρεία Sony εμφανίζεται μόνο μία φορά στο αποτέλεσμα.

### **Η Λέξη Κλειδί Order By**

Η λέξη κλειδί ORDER BY χρησιμοποιείται για να ταξινομήσει το αποτέλεσμα. Το ORDER BY clause χρησιμοποιείται για να ταξινομήσει τις γραμμές.

Πίνακας Παραγγελίες :

Company	OrderNumber
Digital Shop	3412
ABC Shop	5678
Sony	2312

Sony	6798
------	------

#### Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αλφαβητική σειρά :

*SELECT Company, OrderNumber FROM Orders*

*ORDER BY Company*

Αποτέλεσμα :

Company	OrderNumber
ABC Shop	5678
Digital Shop	3412
Sony	6798
Sony	2312

#### Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αλφαβητική σειρά ΚΑΙ (AND) τις παραγγελίες σε αριθμητική σειρά :

*SELECT Company, OrderNumber FROM Orders*

*ORDER BY Company, OrderNumber*

Αποτέλεσμα :

Company	OrderNumber
ABC Shop	5678
Digital Shop	3412
Sony	2312
Sony	6798

#### Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αντίστροφη αλφαβητική σειρά (reverse alphabetical order) :

*SELECT Company, OrderNumber FROM Orders*

*ORDER BY Company DESC*



Αποτέλεσμα :

Company	OrderNumber
Sony	2312
Sony	6798
Digital Shop	3412
ABC Shop	5678

### Άσκηση σε SQL

Δημιουργήστε έναν πίνακα με όνομα Customers σε μια βάση δεδομένων με τις εξής στήλες :

CompanyName	ContactName	Address	City
-------------	-------------	---------	------

Καταχωρείστε στοιχεία στον πίνακα και δοκιμάστε τις εξής εντολές της SQL :

```
SELECT * FROM customers
```

```
SELECT CompanyName, ContactName
```

```
FROM customers
```

```
SELECT * FROM customers
```

```
WHERE companyname LIKE 'α%'
```

```
SELECT CompanyName, ContactName
```

```
FROM customers
```

```
WHERE CompanyName > 'γ'
```

```
AND ContactName > 'γ'
```

### Η Εντολή INSERT INTO

Η εντολή INSERT INTO εισάγει νέες γραμμές σ' έναν πίνακα. Η σύνταξή της είναι ως εξής :

```
INSERT INTO όνομα_πίνακα
```

```
VALUES (τιμή1, τιμή2, ...)
```

Μπορούμε επίσης να καθορίσουμε τις στήλες για τις οποίες θέλουμε να εισάγουμε δεδομένα :

```
INSERT INTO όνομα_πίνακα(στήλη1, στήλη2, ...)
```

```
VALUES (τιμή1, τιμή2, ...)
```

Ο επόμενος πίνακας "Persons" :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα

και αυτή η εντολή SQL :

*INSERT INTO Persons*

*VALUES ('Σιάμκουρης', 'Ιωάννης', 'Π. Μελά 90', 'Καστοριά')*

δίνουν αυτό το αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

Εισαγωγή Δεδομένων σε Συγκεκριμένες Στήλες

Ο επόμενος πίνακας "Persons" :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

και η επόμενη εντολή SQL :

*INSERT INTO Persons (LastName, Address)*

*VALUES ('Νικολάου', 'Ταγμ. Ναούμ 30')*

δίνουν αυτό το αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου		Ταγμ. Ναούμ 30	

## Η Εντολή Update

Η εντολή UPDATE ενημερώνει ή αλλάζει γραμμές. Η σύνταξή της είναι ως εξής :

**UPDATE** όνομα\_πίνακα **SET** όνομα\_στήλης=νέα\_τιμή

**WHERE** όνομα\_στήλης=τιμή

Αρχικός πίνακας Person :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου		Ταγμ. Ναούμ 30	

### Ενημέρωση μίας Στήλης σε μια Γραμμή

Θέλουμε να προσθέσουμε ένα όνομα στο άτομο που έχει το επώνυμο "Νικολάου" :

**UPDATE** Person **SET** FirstName = 'Αθηνά'

**WHERE** LastName = 'Νικολάου'

### Ενημέρωση Πολλών Στηλών σε μια Γραμμή

Για το ίδιο άτομο θέλουμε να αλλάξουμε τη διεύθυνση και να προσθέσουμε ένα όνομα για την πόλη :

**UPDATE** Person

**SET** Address = 'Μεγαρόβου 12', City = 'Φλώρινα'

**WHERE** LastName = 'Νικολάου'

Αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

## Η Εντολή Delete

Η εντολή DELETE χρησιμοποιείται για να διαγράψουμε γραμμές από έναν πίνακα. Η σύνταξή της είναι ως εξής :

**DELETE FROM** όνομα\_πίνακα

**WHERE** όνομα\_στήλης = τιμή

Αρχικός πίνακας Person :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

### Διαγραφή μιας Γραμμής

Θα διαγράψουμε την "Νικολάου Αθηνά" :

**DELETE FROM Person WHERE LastName = 'Νικολάου'**

Αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

## Οι Συναρτήσεις Count της SQL

Η SQL έχει ενσωματωμένες συναρτήσεις για τη μέτρηση (counting) των εγγραφών μιας βάσης δεδομένων.

Η σύνταξη για τις ενσωματωμένες συναρτήσεις COUNT είναι η εξής :

**SELECT COUNT(στήλη) FROM** πίνακας

Η Συνάρτηση COUNT(\*)

Η συνάρτηση COUNT(\*) επιστρέφει τον αριθμό των επιλεγμένων γραμμών από μια επιλογή (selection).

Μ' αυτόν τον πίνακα "Persons" :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	19

Το επόμενο παράδειγμα επιστρέφει τον αριθμό των γραμμών του πίνακα :

*SELECT COUNT(\*) FROM Persons*

Αποτέλεσμα : 3

Το επόμενο παράδειγμα επιστρέφει τον αριθμό των ατόμων που είναι πάνω από 20 χρονών :

*SELECT COUNT(\*) FROM Persons where Age>20*

Αποτέλεσμα : 2

Η Συνάρτηση COUNT(column)

Η συνάρτηση COUNT(column) επιστρέφει τον αριθμό των γραμμών χωρίς τιμή NULL στη συγκεκριμένη στήλη.

Μ' αυτόν τον πίνακα "Persons" :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	

Το επόμενο παράδειγμα βρίσκει τον αριθμό των ατόμων που έχουν τιμή στο πεδίο "Age" του πίνακα "Persons" :

*SELECT COUNT(Age) FROM Persons*

Αποτέλεσμα : 2

Η συνάρτηση COUNT(column) είναι χρήσιμη για να βρίσκουμε τις στήλες που δεν έχουν τιμή. Το αποτέλεσμα είναι κατά ένα λιγότερο από τον αριθμό των γραμμών του αρχικού πίνακα επειδή ένα από τα άτομα δεν έχει τιμή στο πεδίο age.

## Οι Λέξεις Κλειδιά COUNT και DISTINCT

Οι λέξεις κλειδιά DISTINCT και COUNT μπορούν να χρησιμοποιηθούν μαζί για να μετρήσουμε τον αριθμό των διακριτών αποτελεσμάτων (distinct results).

Η σύνταξη είναι ως εξής :

***SELECT DISTINCT COUNT(στήλες) FROM πίνακας***

Με τον επόμενο πίνακα "Orders" :

Company	OrderNumber
Hitachi	3412
Sony	2312
ABC	4678
Sony	6798

Η επόμενη εντολή SQL :

***SELECT COUNT(Company) FROM Orders***

θα δώσει αυτό το αποτέλεσμα : 4

Η επόμενη εντολή SQL :

***SELECT DISTINCT COUNT(Company) FROM Orders***

θα δώσει αυτό το αποτέλεσμα : 3

## Οι Συναρτήσεις της SQL

Η SQL έχει πολλές ενσωματωμένες συναρτήσεις για να μπορούμε να κάνουμε μετρήσεις (counting) και υπολογισμούς (calculations).

Η γενική σύνταξη για τις ενσωματωμένες συναρτήσεις της SQL είναι η εξής :

***SELECT function(στήλη) FROM πίνακας***

Ο αρχικός πίνακας :

Name	Age
Αντωνιάδης Ηλίας	34

Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	19

#### Η Συνάρτηση AVG(column)

Η συνάρτηση AVG επιστρέφει τη μέση τιμή μιας στήλης σε μια επιλογή. Οι τιμές NULL δεν περιλαμβάνονται στους υπολογισμούς.

Το επόμενο παράδειγμα επιστρέφει τον μέσο όρο ηλικίας των ατόμων που υπάρχουν στον πίνακα "Persons" :

```
SELECT AVG(Age) FROM Persons
```

Αποτέλεσμα : 32.67

Το επόμενο παράδειγμα επιστρέφει τον μέσο όρο ηλικίας των ατόμων που είναι πάνω από 20 χρονών :

```
SELECT AVG(Age) FROM Persons where Age>20
```

Αποτέλεσμα : 39.5

#### Η Συνάρτηση MAX(column)

Η συνάρτηση MAX επιστρέφει την μεγαλύτερη τιμή μιας στήλης. Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

#### Παράδειγμα

```
SELECT MAX(Age) FROM Persons
```

Αποτέλεσμα : 45

#### Η Συνάρτηση MIN(column)

Η συνάρτηση MIN επιστρέφει την μικρότερη τιμή μιας στήλης. Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

#### Παράδειγμα

```
SELECT MIN(Age) FROM Persons
```

Αποτέλεσμα : 19

Οι συναρτήσεις MIN και MAX μπορούν επίσης να χρησιμοποιηθούν σε στήλες που περιέχουν κείμενο (text), για να βρούμε την μεγαλύτερη ή μικρότερη τιμή σε αλφαβητική σειρά.

#### Η Συνάρτηση SUM(column)

Η συνάρτηση SUM επιστρέφει το άθροισμα μιας στήλης για μια συγκεκριμένη επιλογή (selection). Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

### Παράδειγμα

Το παρακάτω παράδειγμα επιστρέφει το άθροισμα όλων των ηλικιών του πίνακα "person" :

```
SELECT SUM(Age) FROM Persons
```

Αποτέλεσμα : 98

### Παράδειγμα

Το παρακάτω παράδειγμα επιστρέφει το άθροισμα των ηλικιών του πίνακα "person" για τα άτομα που είναι πάνω από 20 χρονών :

```
SELECT SUM(Age) FROM Persons where Age>20
```

Αποτέλεσμα : 79

### 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 κ.ο.κ.

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 € κ.ο.κ. 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),
Ημηνία_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';

## **Η Λέξη Κλειδί HAVING**

Η λέξη κλειδί HAVING έχει προστεθεί στην SQL επειδή η λέξη κλειδί WHERE δεν μπορεί να χρησιμοποιηθεί σε αθροιστικές συναρτήσεις, όπως είναι η SUM.

Η σύνταξη της συνάρτησης HAVING είναι η εξής :

**SELECT** στήλη, **SUM**(στήλη) **FROM** πίνακας

**GROUP BY** στήλη

**HAVING SUM**(στήλη) συνθήκη

Ο παρακάτω πίνακας "Sales" :

Company	Amount
Grundig	5500
IBM	4500
Grundig	7100

και αυτή η εντολή SQL :

*SELECT Company, SUM(Amount) FROM Sales*

*GROUP BY Company HAVING SUM(Amount)>10000*

θα δώσουν αυτό το αποτέλεσμα :

Company	SUM(Amount)
Grundig	12600

### **Τα Ψευδώνυμα (Aliases)**

Στην SQL, τα ψευδώνυμα (aliases) χρησιμοποιούνται για ονόματα στηλών και πινάκων.

#### **Ψευδώνυμο Στήλης (Column Name Alias)**

Η σύνταξη είναι ως εξής :

***SELECT** στήλη **AS** column\_alias **FROM** πίνακας*

#### **Ψευδώνυμο Πίνακα (Table Name Alias)**

Η σύνταξη είναι ως εξής :

***SELECT** στήλη **FROM** πίνακας **AS** table\_alias*

#### **Παράδειγμα με Column Alias**

Ο επόμενος πίνακας Persons :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα

Σιάγκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

και η εξής εντολή SQL :

```
SELECT LastName AS Family, FirstName AS Name
FROM Persons
```

δίνουν αυτό το αποτέλεσμα :

<b>Family</b>	<b>Name</b>
Σουμπάση	Μαρία
Σιάγκουρης	Ιωάννης
Νικολάου	Αθηνά

Παράδειγμα με Table Alias

Ο επόμενος πίνακας Persons :

<b>LastName</b>	<b>FirstName</b>	<b>Address</b>	<b>City</b>
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάγκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

και η εξής εντολή SQL :

```
SELECT LastName, FirstName
FROM Persons AS Employees
```

δίνουν αυτό το αποτέλεσμα :

<b>LastName</b>	<b>FirstName</b>
Σουμπάση	Μαρία
Σιάγκουρης	Ιωάννης
Νικολάου	Αθηνά

## Ενωση Πινάκων (Join)

Μερικές φορές πρέπει να επιλέξουμε δεδομένα από δύο πίνακες για να δημιουργήσουμε ένα πιο πολύπλοκο αποτέλεσμα. Θα πρέπει να κάνουμε μια *ένωση* (*join*).

Οι πίνακες μιας βάσης δεδομένων μπορούν να συσχετιστούν μεταξύ τους με *κλειδιά* (*keys*). Ένα *πρωτεύον κλειδί* (*primary key*) είναι μια στήλη με μια μοναδική τιμή στην κάθε γραμμή. Ο σκοπός είναι να ενώσει τα δεδομένα μαζί από διάφορους πίνακες, χωρίς να έχουμε επανάληψη όλων των δεδομένων σε κάθε πίνακα.

Στον πίνακα "Employees" παρακάτω, η στήλη "ID" είναι το πρωτεύον κλειδί (*primary key*), που σημαίνει ότι δεν μπορούν να υπάρχουν δύο γραμμές που να έχουν το ίδιο ID. Το ID είναι αυτό που ξεχωρίζει δύο άτομα ακόμη κι αν έχουν το ίδιο όνομα.

Στους πίνακες παραδειγμάτων παρακάτω, προσέχουμε τα εξής :

- Η στήλη "ID" είναι το πρωτεύον κλειδί του πίνακα "Employees".
- Η στήλη "ID" του πίνακα "Orders" χρησιμοποιείται για να αναφερόμαστε στα άτομα του πίνακα "Employees", χωρίς να χρησιμοποιούμε τα ονόματά τους.

### Πίνακας Employees :

ID	Name
01	Νικολάου Αθηνά
02	Γεωργιάδης Ηλίας
03	Παπαδόπουλος Στέφανος
04	Σουμπάσης Ιωάννης

### Πίνακας Orders :

ID	Product
01	Printer
03	Computer
03	Scanner

### Παράδειγμα

Ποιοι έχουν παραγγείλει προϊόντα και τι έχουν παραγγείλει;

*SELECT Employees.Name, Orders.Product*

*FROM Employees, Orders*

*WHERE Employees.ID = Orders.ID*

Αποτέλεσμα :

Name	Product
Νικολάου Αθηνά	Printer
Παπαδόπουλος Στέφανος	Computer
Παπαδόπουλος Στέφανος	Scanner

### Παράδειγμα

Ποιοι έχουν παραγγείλει εκτυπωτή (printer);

*SELECT Employees.Name*

*FROM Employees, Orders*

*WHERE Employees.ID = Orders.ID*

*AND Orders.Product = 'Printer'*

Αποτέλεσμα :

Name
Νικολάου Αθηνά

### 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 SYMMETOXH*

*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 (Ενωση)**

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 SYMMETOXH*

*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 SYMMETOXH*

*Where Κωδικός\_Αγώνα = 302);*

## Οι Όροι In και Not In

Οι εντολές της SQL έχουν την ιδιότητα να εφαρμόζονται σε σχέσεις και το αποτέλεσμα 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 ( $\square$ ). 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 ( $\square$ ). 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 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;***

## **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 διαγράψουμε 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;***