code4sp
coding for social promotion

Με συγχρηματοδότηση από το
πρόγραμμα «Erasmus+»
της Ευρωπαϊκής Ένωσης

4.1.:

Πακέτο Εκπαιδευτικού Υλικού του έργου

WP3:

Εκπαιδευτικό Υλικό του έργου Code4SP

Από:

Με συγχρηματοδότηση από το πρόγραμμα «Erasmus+» της Ευρωπαϊκής Ένωσης

Πληροφορίες σχετικά με το έργο

Ακρωνύμιο του έργου: Code4SP

Τίτλος του έργου: Coding for Social Promotion

Αριθμός Έργου: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN

Ιστοσελίδα του έργου: www.code4sp.eu

Εταίρος Συντάκτης: C.I.P. Citizens in Power

Έκδοση Εγγράφου: 1

Ημερομηνία Παραγωγής: 11/03/2022

Ιστορικό Εγγράφου			
Ημερομηνία	Έκδοση	Συντάκτης	Περιγραφή
11/03/2022	1	CIP	Προσχέδιο

Περιεχόμενα

Ενότητα:

4. Γλώσσα SQL

Προαπαιτούμενες γνώσεις:

Βασικές γνώσεις υπολογιστών, εγκατεστημένο βασικό λογισμικό και βασικές γνώσεις εργασίας με αρχεία.

Φόρτος εργασίας:

10 ώρες.

Περιγραφή:

Σε αυτή την ενότητα, θα καλύψουμε τα βασικά της SQL για να εξοικειώσουμε τους μαθητές με τον κόσμο του προγραμματισμού και να τους ενθαρρύνουμε να αποκτήσουν περισσότερη εμπειρία στην SQL. Εξηγούμε τα χαρακτηριστικά, τη σύνταξη και άλλους σχετικούς όρους που μπορεί να έχουν ακούσει ή να είναι εξοικειωμένοι οι εκπαιδευόμενοι και πώς αυτά ταιριάζουν στη γλώσσα προγραμματισμού. Παρέχουμε μια λεπτομερή περιγραφή της λογικής και της σύνταξης που χρησιμοποιούνται στην SQL, τη δομή της και άλλες βασικές και σημαντικές λειτουργίες.

Μαθησιακά αποτελέσματα:

- Αναγνώριση της έννοιας και της χρήσης της Structured Query Language (SQL) στα σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)
- Διαμόρφωση βασικής σύνταξης ερωτημάτων SQL
- Χρήση της SQL για να της απόκτηση πρόσβασης, τη διαχείριση και τον χειρισμό των ΣΔΒΔ

Απαιτούμενα υλικά:

- Υπολογιστής ή φορητός υπολογιστής
- Σύνδεση στο Διαδίκτυο
- Διαδικτυακό πρόγραμμα μεταγλώττισης της SQL (https://www.mycompiler.io/new/sql)
- Διαδικτυακό πρόγραμμα επεξεργασίας κειμένου (https://www.w3schools.com/html/default.asp)

Σενάριο μαθήματος:

Ο συνολικός χρόνος που αφιερώνεται σε αυτή την ενότητα είναι 10 ώρες και εναπόκειται στην κρίση του εκπαιδευτικού να αποφασίσει πόσος χρόνος θα δαπανηθεί για κάθε υποενότητα. Προτείνουμε να χρησιμοποιήσετε τις παρουσιάσεις κατάρτισης PPT που δημιουργήθηκαν ρητά για αυτό το θέμα για να διευκολύνετε τη διαδικασία διδασκαλίας και να αυξήσετε την αποδοτικότητα του χρόνου. Αυτές οι παρουσιάσεις περιλαμβάνουν τα ακόλουθα:

- Προοδευτική ανάπτυξη επιμέρους θεμάτων και βασικών εννοιών για τη διατήρηση, και
- Προτεινόμενες ασκήσεις.

Ανάλογα με τις προτιμήσεις του εκπαιδευτικού, η προοδευτική ανάπτυξη των παρουσιάσεων επιτρέπει την ολοκλήρωση του μαθήματος πάνω στην SQL εντός του καθορισμένου χρόνου, δηλαδή 10 ώρες. Οι παρουσιάσεις μπορούν επίσης να διατεθούν στους εκπαιδευόμενους για αυτοδιδασκαλία.

Υποενότητες:

4.1. Τα βασικά της SQL

4.2. Βάσεις δεδομένων SQL

4.3. Παραπομπές SQL

4.4 Παραδείγματα SQL

Επιπλέον πόροι:

- W3Schools - Οδηγός για κάθε λέξη-κλειδί και λειτουργία SQL, και

παραδείγματα για καθένα από αυτά

- Tutorialspoint – Ένας άλλος λεπτομερής οδηγός για τις λέξεις-κλειδιά και τις λειτουργίες SQL, και διάφορα παραδείγματα για κάθε μία από αυτές

4.1. Τα βασικά της SQL

Τι είναι η SQL;

Το ακρωνύμιο SQL σημαίνει Structured Query Language (Δομημένη Γλώσσα Ερωτημάτων). Η SQL είναι μια τυποποιημένη γλώσσα προγραμματισμού ειδικά σχεδιασμένη για την αποθήκευση, ανάκτηση, διαχείριση ή χειρισμό δεδομένων που βρίσκονται μέσα σε ένα σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ). Μια σχεσιακή βάση δεδομένων είναι μια συλλογή στοιχείων δεδομένων με προκαθορισμένες σχέσεις μεταξύ τους. Τα στοιχεία αυτά οργανώνονται ως ένα σύνολο πινάκων με στήλες και σειρές.

Η SQL εντάχθηκε στο πρότυπο ISO το 1987.

Η SQL είναι η πιο ευρέως εφαρμοσμένη γλώσσα βάσης δεδομένων που υποστηρίζεται από δημοφιλή συστήματα σχεσιακών βάσεων δεδομένων, όπως MySQL, SQL Server, και Oracle. Η SQL αναπτύχθηκε αρχικά στην IBM στις αρχές της δεκαετίας του 1970. Αρχικά, ονομαζόταν SEQUEL (Structured English Query Language) και αργότερα άλλαξε σε SQL (προφέρεται ως S-Q-L).

Εφαρμογές της SQL

Η SQL είναι μία από τις πιο ευρέως χρησιμοποιούμενες γλώσσες ερωτημάτων για βάσεις δεδομένων. Μερικές από τις πολλές εφαρμογές της είναι:

- Να επιτρέπει στους χρήστες να **έχουν πρόσβαση σε δεδομένα** στα σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων.
- Να επιτρέπει στους χρήστες να **περιγράψουν τα δεδομένα**.
- Να επιτρέπει στους χρήστες να **ορίσουν τα δεδομένα** σε μια βάση δεδομένων και να χειρίζονται τα δεδομένα αυτά.

Σύνταξη στην SQL

Οι εντολές στην SQL είναι απλές, σαν να γράφουμε προτάσεις, αλλά με συγκεκριμένη σύνταξη.

Μια εντολή στην SQL αποτελείται από μια ακολουθία λέξεων-κλειδιών, αναγνωριστικών κ.λπ., που τερματίζονται από ένα ερωτηματικό (;).

Παράδειγμα:

SELECT όνομα_εργαζομένου, ημερομηνία_πρόσληψης, μισθός FROM Εργαζόμενοι WHERE μισθός > 5000;

Για πιο εύκολη αναγνωσιμότητα, μπορείτε να γράψετε την ίδια εντολή ως εξής:

SELECT όνομα_εργαζομένου, ημερομηνία_πρόσληψης, μισθός

FROM Εργαζόμενοι

WHERE μισθός > 5000;

Χρησιμοποιήστε το ερωτηματικό του ελληνικού αλφαβήτου στο τέλος μιας εντολής SQL, καθώς τερματίζει τη εντολή ή υποβάλλει τις πληροφορίες στον διακομιστή βάσης δεδομένων.

Διάκριση πεζών-κεφαλαίων στην SQL

Εξετάστε μια άλλη εντολή στην SQL που ανακτά εγγραφές από τον πίνακα Εργαζομένων:

SELECT όνομα_εργαζομένου, ημερομηνία_πρόσληψης, μισθός FROM Εργαζόμενοι;

Η ίδια εντολή μπορεί επίσης να γραφτεί ως εξής:

select όνομα_εργαζομένου, ημερομηνία_πρόσληψης, μισθός from εργαζόμενοι;

Οι λέξεις-κλειδιά στην SQL είναι χωρίς διάκριση πεζών-κεφαλαίων, που σημαίνει ότι η εντολή SELECT είναι το ίδιο πράγμα με το select.

Ωστόσο, η βάση δεδομένων και τα ονόματα των πινάκων μπορεί να είναι, σύμφωνα με την περίπτωση, ανάλογα με το λειτουργικό σύστημα. Σε γενικές γραμμές, οι πλατφόρμες Unix ή Linux είναι ευαίσθητες στη διάκριση πεζών-κεφαλαίων, ενώ οι πλατφόρμες Windows όχι.

Επιλογή δεδομένων στην SQL (SELECT)

Η εντολή SELECT επιλέγει ή ανακτά δεδομένα από έναν ή περισσότερους πίνακες. Μπορείτε να χρησιμοποιήσετε αυτή την εντολή για να ανακτήσετε όλες τις σειρές από έναν πίνακα με μία κίνηση ή να ανακτήσετε μόνο εκείνες τις σειρές που ικανοποιούν μια συγκεκριμένη συνθήκη ή έναν συνδυασμό συνθηκών.

Ας υποθέσουμε ότι έχουμε έναν πίνακα με το όνομα Εργαζόμενοι στη βάση δεδομένων μας που περιέχει τις ακόλουθες εγγραφές:

| κωδικός_εργαζομένου| όνομα_εργαζομένου| ημερομηνία_πρόσληψης| μισθός| κωδικός_τμήματος|

+--------+--------------+------------+--------+---------+

| 1 | Ίθαν Χαντ | 2001-05-01 | 5000 | 4 |

| 2 | Τόνι Μοντάνα | 2002-07-15 | 6500 | 1 |

| 3 | Σάρα Κόνορ | 2005-10-18 | 8000 | 5 |

| 4 | Ρικ Ντέκαρντ | 2007-01-03 | 7200 | 3 |

| 5 | Μάρτιν Μπλανκ | 2008-06-24 | 5600 | NULL |

+--------+--------------+------------+--------+---------+

Επιλογή όλων από έναν πίνακα

Η ακόλουθη εντολή θα επιλέξει όλες τις σειρές από τον πίνακα των εργαζομένων.

>> SELECT * FROM Εργαζόμενοι;

Επιλογή συγκεκριμένων στηλών από τον πίνακα:

Εάν δεν χρειάζεστε όλα τα δεδομένα, μπορείτε να επιλέξετε συγκεκριμένες στήλες, όπως φαίνεται παρακάτω:

SELECT ταυτότητα_εργαζομένου, όνομα_εργαζομένου, ημερομηνία_πρόσληψης, μισθός

FROM εργαζόμενοι;

Μετά την εκτέλεση της παραπάνω εντολής, θα λάβετε μια απόδοση όπως αυτή:

| κωδικός_εργαζομένου| όνομα_εργαζομένου| ημερομηνία_πρόσληψης| μισθός|

+--------+--------------+------------+--------+

| 1 | Ίθαν Χαντ | 1995-10-30 | 5000 |

| 2 | Τόνι Μοντάνα | 1990-07-15 | 6500 |

| 3 | Σάρα Κόνορ | 2011-04-13 | 5600 |

| 4 | Ρικ Ντέκαρντ | 2005-10-18 | 7200 |

| 5 | Μάρτιν Μπλανκ | 1996-05-24 | 8000 |

+--------+--------------+------------+--------+

Επιλογή ορισμένων δεδομένων στην SQL (SELECT DISTINCT)

Η εντολή SELECT DISTINCT παραλείπει τις διπλές τιμές όταν χρησιμοποιείται σε ένα ερώτημα.

Μπορεί να βρείτε διπλές τιμές μέσα σε έναν πίνακα, αλλά μερικές φορές μπορεί να θέλετε να δείτε τις «μοναδικές» τιμές.

Σύνταξη:

SELECT DISTINCT στήλη1, στήλη2, ...
FROM όνομα_πίνακα;

Στα παρακάτω παραδείγματα, θα χρησιμοποιήσουμε τον πίνακα Πελατών που περιέχει δεδομένα σχετικά με τους πελάτες μας.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujilo Emparedados y helados	Ana Trujilo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Πίνακας πελατών - ΠΑΡΑΔΕΙΓΜΑ ΤΗΣ ΕΝΤΟΛΗΣ SELECT DISTINCT

(Πηγή: https://www.w3schools.com/sql/sql_distinct.asp)

Για να *επιλέξετε όλες τις τιμές* από τη στήλη Χώρα στον πίνακα Πελάτες, θα χρησιμοποιήσετε την ακόλουθη εντολή:
SELECT Χώρα FROM Πελάτες;

Ωστόσο, η εντολή αυτή θα περιλαμβάνει διπλές τιμές.

Εάν θέλετε να παραλείψετε τις διπλότυπες τιμές από το ερώτημά σας, χρησιμοποιήστε την SELECT DISTINCT:
SELECT DISTINCT Χώρα FROM Πελάτες;

Ας υποθέσουμε ότι θέλετε να απαριθμήσετε τον αριθμό των διαφορετικών χωρών των πελατών. Θα χρησιμοποιήσετε την ακόλουθη εντολή:
SELECT COUNT(DISTINCT Χώρα) FROM Πελάτες;

Λάβετε υπόψη ότι αυτό το παράδειγμα δεν θα λειτουργήσει στο Firefox, καθώς το COUNT(DISTINCT όνομα_στήλης) δεν υποστηρίζεται στο MS Access.

Για να λάβετε το ισοδύναμο αποτέλεσμα στο σύστημα Access, χρησιμοποιήστε αυτό:

```
>> SELECT Count(*) AS DistinctCountries
    FROM (SELECT DISTINCT Χώρα FROM Πελάτες);
```

Όρος WHERE στην SQL

Προηγουμένως, μάθαμε πώς να επιλέγουμε όλα τα αρχεία από έναν πίνακα ή στήλες ενός πίνακα.

Ωστόσο, σε πραγματικές περιπτώσεις, πρέπει γενικά να επιλέξουμε, να ενημερώσουμε ή να διαγράψουμε μόνο εκείνα τα αρχεία που ικανοποιούν καθορισμένες συνθήκες, όπως οι χρήστες που ανήκουν σε μια συγκεκριμένη ηλικιακή ομάδα, χώρα κ.λπ.

Ο όρος WHERE χρησιμοποιείται με τις εντολές SELECT, UPDATE, και DELETE.

Ο όρος WHERE χρησιμοποιείται με την εντολή SELECT για την εξαγωγή μόνο εκείνων των εγγραφών που ικανοποιούν καθορισμένες συνθήκες.

Σύνταξη:

```
SELECT στήλη_λίστα
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Τώρα, ας δούμε μερικά παραδείγματα που δείχνουν πώς λειτουργεί.

Ας υποθέσουμε ότι έχουμε έναν πίνακα που ονομάζεται Εργαζόμενοι στη βάση δεδομένων μας με τα ακόλουθα αρχεία:

| κωδικός_εργαζομένου| όνομα_εργαζομένου| ημερομηνία_πρόσληψης| μισθός| κωδικός_τμήματος|

+--------+--------------+------------+--------+---------+

| 1 | Ίθαν Χαντ | 2001-05-01 | 5000 | 4 |

| 2 | Τόνι Μοντάνα | 2002-07-15 | 6500 | 1 |

| 3 | Σάρα Κόνορ | 2005-10-18 | 8000 | 5 |

| 4 | Ρικ Ντέκαρντ | 2007-01-03 | 7200 | 3 |

| 5 | Μάρτιν Μπλανκ | 2008-06-24 | 5600 | NULL |

+--------+--------------+------------+--------+---------+

Η ακόλουθη εντολή της SQL θα επιλέξει όλους τους υπαλλήλους από τον πίνακα *των Εργαζομένων των οποίων ο μισθός είναι μεγαλύτερος από 7000*:
SELECT * FROM εργαζόμενοι WHERE μισθός > 7000;

Ο όρος WHERE απλώς φιλτράρει τα ανεπιθύμητα δεδομένα.

Ένα άλλο παράδειγμα θα ήταν να *επιλέξετε όλους* τους υπαλλήλους με *κωδικό τμήματος =1*:
SELECT * FROM εργαζόμενοι WHERE κωδικό_τμήματος=1;

Ο ακόλουθος πίνακας παρουσιάζει τους τελεστές που μπορούν να χρησιμοποιηθούν με τον όρο WHERE:

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

Πίνακας των τελεστών που χρησιμοποιούνται με τον όρο WHERE

(Πηγή: https://www.w3schools.com/sql/sql_where.asp)

Οι τελεστές AND, OR και NOT στην SQL

Ο όρος WHERE μπορεί να συνδυαστεί με τους τελεστές AND, OR, και NOT.

Οι τελεστές AND και OR χρησιμοποιούνται για το φιλτράρισμα εγγραφών με βάση περισσότερες από μία συνθήκες.

Ο τελεστής **AND** εμφανίζει μια εγγραφή εάν όλες οι συνθήκες που χρησιμοποιούν AND είναι TRUE.

Σύνταξη:

SELECT στήλη1, στήλη2, ...

FROM όνομα_πίνακα

WHERE συνθήκη1 AND συνθήκη2 AND συνθήκη3...;

Παράδειγμα: Επιλέξτε όλα τα πεδία από τον πίνακα πελατών όπου η χώρα είναι η «Γερμανία» ΚΑΙ η πόλη είναι το «Βερολίνο».

SELECT * FROM Πελάτες

WHERE Χώρα='Γερμανία' AND Πόλη='Βερολίνο';

Ο τελεστής **OR** εμφανίζει μια εγγραφή εάν οποιαδήποτε από τις συνθήκες που χρησιμοποιούν το OR είναι TRUE.

Σύνταξη:

```
SELECT στήλη1, στήλη2, ...
FROM όνομα_πίνακα
WHERE συνθήκη1 OR συνθήκη2 OR συνθήκη3...;
```

Παράδειγμα 1: Επιλέξτε όλα τα πεδία από τον πίνακα πελατών όπου η πόλη είναι το «Βερολίνο» ή το «Μόναχο»

```
SELECT * FROM Πελάτες
WHERE Πόλη = ‘Βερολίνο' OR Πόλη = ‘Μόναχο';
```

Παράδειγμα 2: Επιλέξτε όλα τα πεδία από τον πίνακα πελατών όπου η χώρα είναι «Γερμανία» ή «Ισπανία».

```
SELECT * FROM Πελάτες

WHERE Χώρα=‘Γερμανία' OR Χώρα=‘Ισπανία';
```

Ο **τελεστής NOT** εμφανίζει μια εγγραφή εάν οι συνθήκες είναι NOT TRUE.

Σύνταξη:

```
SELECT στήλη1, στήλη2, ...
FROM όνομα_πίνακα
WHERE NOT συνθήκη;
```

Παράδειγμα: Επιλέξτε όλα τα πεδία από τον πίνακα πελατών όπου η χώρα ΔΕΝ είναι η «Γερμανία».

```
SELECT * FROM Πελάτες
WHERE NOT Χώρα=‘Γερμανία';
```

Συνδυασμός των AND, OR και NOT

Παράδειγμα 1: Επιλέξτε όλες τις σειρές από τον πίνακα Πελάτες όπου η χώρα είναι η Γερμανία και η πόλη πρέπει να είναι είτε το Βερολίνο είτε το Μόναχο.

```
SELECT * FROM Πελάτες
WHERE Χώρα='Γερμανία' AND (Πόλη='Βερολίνο' OR Πόλη='Μόναχο');
```

Παράδειγμα 2: Επιλέξτε όλες τις σειρές από τον πίνακα Πελάτες όπου η χώρα ΔΕΝ είναι η Γερμανία και ούτε οι ΗΠΑ.

```
SELECT * FROM Πελάτες
WHERE NOT Χώρα='Γερμανία' AND NOT Χώρα='ΗΠΑ';
```

Ταξινόμηση στην SQL (ORDER BY)

Η λέξη-κλειδί ORDER BY ταξινομεί το σύνολο των αποτελεσμάτων κατά αύξουσα ή φθίνουσα σειρά. Η λέξη-κλειδί ORDER BY ταξινομεί τις εγγραφές με αύξουσα σειρά από προεπιλογή.
Για να ταξινομήσετε τις εγγραφές με φθίνουσα σειρά, χρησιμοποιήστε τη λέξη-κλειδί DESC.

Σύνταξη:

```
SELECT στήλη1, στήλη, ...
FROM όνομα_πίνακα
ORDER BY στήλη1, στήλη2, ... ASC|DESC;
```

Στα παρακάτω παραδείγματα, θα χρησιμοποιήσουμε τις ελληνικές αντιστοιχίες από τον παρακάτω πίνακα Πελατών:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

ΠΙΝΑΚΑΣ ΠΕΛΑΤΩΝ – ΠΑΡΑΔΕΙΓΜΑ ΤΗΣ ΕΝΤΟΛΗΣ ORDER BY
(Πηγή: https://www.w3schools.com/sql/sql_orderby.asp)

Παράδειγμα 1: Επιλέγει όλους τους πελάτες από τον πίνακα Πελατών και τους ταξινομεί κατά τη στήλη Χώρα

```
SELECT * FROM Πελάτες
ORDER BY Χώρα;
```

Παράδειγμα 2: Επιλέγει όλους τους πελάτες από τον ίδιο πίνακα και τους ταξινομεί με φθίνουσα σειρά κατά τη στήλη Χώρα

```
SELECT * FROM Πελάτες
ORDER BY Χώρα DESC;
```

Παράδειγμα 3: Επιλέγει όλους τους πελάτες από τον ίδιο πίνακα και τους ταξινομεί ανά χώρα και όνομα πελάτη.

```
SELECT * FROM Πελάτες
ORDER BY Χώρα, ΌνομαΠελάτη;
```

Εδώ, η ταξινόμηση γίνεται αρχικά ανά χώρα. Ωστόσο, εάν υπάρχουν ορισμένες σειρές που έχουν την ίδια χώρα, τότε ταξινομούνται ανά Όνομα Πελάτη.

Παράδειγμα 4: Επιλέγει όλους τους πελάτες από τον ίδιο πίνακα και τους ταξινομεί κατά αύξουσα σειρά ανά χώρα και κατά φθίνουσα σειρά ανά όνομα πελάτη

```
SELECT * FROM Πελάτες
ORDER BY Χώρα ASC, ΌνομαΠελάτη DESC;
```

Εισαγωγή Δεδομένων στην SQL (INSERT INTO)

Η εντολή INSERT INTO εισάγει νέες εγγραφές σε έναν πίνακα.

Για να τρέξει σωστά ο κώδικάς σας, καθορίστε τα ονόματα των στηλών και τις τιμές που θα εισαχθούν.

Σύνταξη:

```
INSERT INTO όνομα_πίνακα (στήλη1, στήλη2, στήλη3, ...)
VALUES (τιμή1, τιμή2, τιμή3, ...);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

(Πηγή: https://www.w3schools.com/sql/sql_insert.asp)

Για παράδειγμα, για να προσθέσετε μια νέα εγγραφή στον πίνακα «Πελάτες», χρησιμοποιήστε τα εξής:

INSERT INTO Πελάτες (ΌνομαΠελάτη, ΌνομαΕπικοινωνίας, Διεύθυνση, Πόλη, ΤαχυδρομικόςΚώδικας, Χώρα)

VALUES ('Κάρντιναλ', 'Τομ Μπ. Έριχσεν', 'Σκάγκεν 21', 'Στάβανγκερ', '4006', 'Νορβηγία');

Κενές τιμές στην SQL (NULL)

Ένα πεδίο με τιμή NULL είναι **ένα πεδίο χωρίς τιμή**. Εάν ένα πεδίο σε έναν πίνακα είναι προαιρετικό, είναι δυνατό να εισαχθεί μια νέα εγγραφή ή να ενημερωθεί μια εγγραφή χωρίς να προστεθεί μια τιμή σε αυτό το πεδίο. Στη συνέχεια, το πεδίο θα αποθηκευτεί με τιμή NULL.

Σημείωση: Μια τιμή NULL διαφέρει από μια μηδενική τιμή ή ένα πεδίο που περιέχει κενά. Ένα πεδίο με τιμή NULL έχει **μείνει κενό** κατά τη δημιουργία εγγραφών!

Έλεγχος για ΤΙΜΕΣ NULL

Είναι αδύνατος ο έλεγχος για τιμές NULL με τελεστές σύγκρισης, όπως =, <, ή <>.

Αντ' αυτού, θα πρέπει να χρησιμοποιήσουμε τους τελεστές **IS NULL** και **IS NOT NULL.**

Σύνταξη *IS NULL*:

```
SELECT όνομα_στήλης
FROM όνομα_πίνακα
WHERE όνομα_στήλης IS NULL;
```

Σύνταξη IS NOT NULL:

```
SELECT όνομα_στήλης
FROM όνομα_πίνακα
WHERE όνομα_στήλης IS NOT NULL;
```

Τα ακόλουθα παραδείγματα χρησιμοποιούν τον ακόλουθο πίνακα:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Πίνακας Πελατών – Παράδειγμα ΤΙΜΩΝ NULL

(Πηγή: https://www.w3schools.com/sql/sql_null_values.asp)

Παράδειγμα του IS NULL

Επιλέγει όλους τους πελάτες με τιμές NULL στη στήλη Διεύθυνση

```
SELECT ΌνομαΠελάτη, ΌνομαΕπικοινωνίας, Διεύθυνση
FROM Πελάτες
WHERE Διεύθυνση IS NULL;
```

Για να αναζητήσετε κενές τιμές, χρησιμοποιείτε πάντα το IS NULL.

Παράδειγμα του IS NOT NULL

Επιλέγει όλους τους πελάτες με τιμές NOT NULL στη στήλη Διεύθυνση

```
SELECT ΌνομαΠελάτη, ΌνομαΕπικοινωνίας, Διεύθυνση
FROM Πελάτες
WHERE Διεύθυνση IS NOT NULL;
```

Ενημέρωση δεδομένων στην SQL (UPDATE)

Η εντολή UPDATE τροποποιεί τις υπάρχουσες εγγραφές ενός πίνακα.

Σύνταξη:

UPDATE όνομα_πίνακα

SET στήλη1 = τίμη1, στήλη2 = τιμή2, ...

WHERE συνθήκη;

Να είστε προσεκτικοί όταν ενημερώνετε εγγραφές σε έναν πίνακα! Παρατηρήστε τον όρο WHERE στην εντολή UPDATE. Ο όρος WHERE προσδιορίζει ποια εγγραφή(-ές) πρέπει να ενημερωθεί (-ούν).

Εάν **παραλείψετε** τον όρο «WHERE», θα ενημερωθούν όλες οι εγγραφές στον πίνακα!

Παράδειγμα

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Πίνακας πελατών – παράδειγμα της εντολής UPDATE
(Πηγή: https://www.w3schools.com/sql/sql_update.asp)

Για να ενημερώσετε τον Κωδικό Πελάτη =1 από τον πίνακα «Πελάτες» στη δειγματική βάση δεδομένων, χρησιμοποιήστε τα ακόλουθα:

UPDATE Πελάτες

SET ΌνομαΕπικοινωνίας = ‘Άλφρεντ Σμιντ’, Πόλη= ‘Φρανκφούρτη'

WHERE ΚωδικόςΠελάτη = 1;

Και ο πίνακας «Πελάτες» θα έχει τώρα την εξής μορφή:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Πίνακας πελατών – παράδειγμα της εντολής UPDATE

(Πηγή: https://www.w3schools.com/sql/sql_update.asp)

Ο όρος **WHERE καθορίζει πόσες εγγραφές** θα ενημερωθούν.

Παράδειγμα: Ενημέρωση της στήλης ΌνομαΕπικοινωνίας σε «Χουάν» για όλες τις εγγραφές όπου η χώρα είναι «Μεξικό» στον πίνακα Πελατών.

UPDATE Πελάτες
SET ΌνομαΕπικοινωνίας=‘Χουάν'
WHERE Χώρα=‘Μεξικό';

Διαγραφή δεδομένων στην SQL (DELETE)

Η εντολή DELETE διαγράφει υπάρχουσες εγγραφές σε έναν πίνακα.
DELETE FROM table_name WHERE condition;

Λάβετε υπόψη ότι πρέπει να είστε προσεκτικοί όταν διαγράφετε καταχωρήσεις σε έναν πίνακα! Παρατηρήστε τον όρο WHERE στην εντολή DELETE. Ο όρος WHERE προσδιορίζει ποια εγγραφή(-ές) πρέπει να διαγραφεί (-ούν).

Εάν **παραλείψετε** τον όρο «WHERE», θα διαγραφούν όλες οι εγγραφές του πίνακα!

Παράδειγμα:

DELETE FROM Πελάτες WHERE ΌνομαΠελάτη='Αλφρεντς Φούτερκιστε';

Διαγραφή όλων των εγγραφών ενός πίνακα

Είναι δυνατή η **διαγραφή όλων των σειρών σε έναν πίνακα χωρίς να διαγραφεί ο ίδιος ο πίνακας**. Αυτό σημαίνει ότι η δομή, τα χαρακτηριστικά και οι δείκτες του πίνακα θα παραμείνουν άθικτα:
DELETE FROM όνομα_πίνακα;

Επιλογή συγκεκριμένου αριθμού δεδομένων στην SQL (SELECT TOP)

Ο όρος SELECT TOP χρησιμοποιείται για να καθορίσει τον αριθμό των εγγραφών που θα επιλεχθούν.

Ο όρος SELECT TOP είναι χρήσιμος σε μεγάλους πίνακες με χιλιάδες εγγραφές. Ωστόσο, η επιλογή μεγάλου αριθμού εγγραφών μπορεί να επηρεάσει την απόδοση.

Σημειώστε ότι δεν υποστηρίζουν όλα τα συστήματα βάσεων δεδομένων τον όρο SELECT TOP. Το MySQL υποστηρίζει τον όρο LIMIT για την επιλογή ενός περιορισμένου αριθμού εγγραφών, ενώ το Oracle χρησιμοποιεί τους όρους FETCH FIRST αριθμός ROWS ONLY και ROWNUM.

Σύνταξη SQL Server/MS Access:

SELECT TOP αριθμός|ποσοστό όνομα(τα)_στήλης(ων)
FROM όνομα_πίνακα
WHERE συνθήκη;

Σύνταξη MySQL:

SELECT όνομα(τα)_στήλης(ων)
FROM όνομα_πίνακα
WHERE συνθήκη
LIMIT αριθμός;
LIMIT number;

Σύνταξη Oracle 12:

SELECT όνομα(τα)_στήλης(ων)
FROM όνομα_πίνακα
ORDER BY όνομα(τα)_στήλης(ων)
FETCH FIRST αριθμός ROWS ONLY;

Σύνταξη σε παλαιότερη έκδοση του Oracle:

SELECT όνομα_στήλης(ών)
FROM όνομα_πίνακα
WHERE ROWNUM <= αριθμός;

Σύνταξη σε παλαιότερη έκδοση του Oracle με την εντολή ORDER BY:

SELECT *
FROM (SELECT όνομα_στήλης(ών) FROM όνομα_πίνακα ORDER BY όνομα_στήλης(ών))
WHERE ROWNUM <=αριθμός;

Θα χρησιμοποιήσουμε τις ελληνικές αντιστοιχίες από τον πίνακα «Πελάτες» που παρουσιάζεται παρακάτω στα ακόλουθα παραδείγματα.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Πίνακας πελατών - Παραδείγματα της εντολής SELECT TOP

(Πηγή: https://www.w3schools.com/sql/sql_top.asp)

Παραδείγματα TOP, LIMIT και FETCH FIRST

Για να επιλέξετε τις τρεις πρώτες εγγραφές από τον πίνακα Πελατών στο SQL Server/MS Access, χρησιμοποιήστε τα παρακάτω:

```
SELECT TOP 3 * FROM Πελάτες;
```

Για να εκτελέσετε ένα ερώτημα με το ίδιο αποτέλεσμα όπως παραπάνω στο MySQL, χρησιμοποιήστε την ακόλουθη εντολή:

```
SELECT * FROM Πελάτες
LIMIT 3;
```

Για να εκτελέσετε ένα ερώτημα με το ίδιο αποτέλεσμα με τα δύο παραπάνω στο Oracle:

```
SELECT * FROM Πελάτες
FETCH FIRST 3 ROWS ONLY;
```

Παραδείγματα Ανώτατου ποσοστού (TOP PERCENT)

Για να επιλέξετε το πρώτο 50% των εγγραφών που βρίσκονται στον πίνακα Πελάτες, εκτελέστε την ακόλουθη εντολή στο SQL Server/MS Access:

SELECT TOP 50 PERCENT * FROM Πελάτες;

Το αντίστοιχό της στο Oracle είναι το ακόλουθο:

SELECT * FROM Πελάτες

FETCH FIRST 50 PERCENT ROWS ONLY;

Παραδείγματα προσθήκης του όρου WHERE

Στο παρακάτω παράδειγμα, επιλέγουμε τις τρεις πρώτες εγγραφές από τον πίνακα Πελατών, όπου η Χώρα είναι η «Γερμανία» σε SQL Server/MS Access:

>> SELECT TOP 3 * FROM Πελάτες

WHERE Χώρα='Γερμανία';

Το αντίστοιχό του στο MySQL:

SELECT * FROM Πελάτες WHERE Χώρα='Γερμανία' LIMIT 3;

Το αντίστοιχό του στο Oracle:

SELECT * FROM Πελάτες

WHERE Χώρα='Γερμανία'

FETCH FIRST 3 ROWS ONLY;

Συναρτήσεις Min και Max στην SQL

Η συνάρτηση MIN() επιλέγει **τις μικρότερες τιμές** από τις επιλεγμένες στήλες.

Σύνταξη της συνάρτησης MIN()

SELECT MIN(όνομα_στήλης)

FROM όνομα_πίνακα

WHERE συνθήκη;

Η συνάρτηση MAX() επιλέγει **τη μεγαλύτερη τιμή** από τις επιλεγμένες στήλες.

Σύνταξη της συνάρτησης MAX()

```
SELECT MAX(όνομα_στήλης)
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Στα παρακάτω παραδείγματα, θα χρησιμοποιήσουμε τον παρακάτω πίνακα με τα Προϊόντα:

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Πίνακας προϊόντων – Παραδείγματα συναρτήσεων Min και Max

(Πηγή: https://www.w3schools.com/sql/sql_min_max.asp)

Παράδειγμα 1: Βρίσκει την τιμή του φθηνότερου προϊόντος

```
SELECT MIN(Τιμή) AS ΦθηνότερηΤιμή
FROM Προϊόντα;
```

Παράδειγμα 2: Βρίσκει την τιμή του ακριβότερου προϊόντος

```
SELECT MAX(Τίμη) AS ΑκριβότερηΤιμή
FROM Προϊόντα;
```

Συναρτήσεις Count, Avg, Sum στην SQL

Η συνάρτηση COUNT() επιλέγει τον αριθμό των σειρών που ταιριάζουν σε ένα καθορισμένο κριτήριο.

Σύνταξη συνάρτησης COUNT()

```
SELECT COUNT(όνομα_στήλης)
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Η συνάρτηση AVG() επιλέγει τη μέση τιμή μιας αριθμητικής στήλης.

Σύνταξη συνάρτησης AVG()

```
SELECT AVG(όνομα_στήλης)
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Η συνάρτηση SUM() επιστρέφει το συνολικό άθροισμα μιας αριθμητικής στήλης.

Σύνταξη συνάρτησης SUM()

```
SELECT SUM(όνομα_στήλης)
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Θα χρησιμοποιήσουμε τον πίνακα Προϊόντα στα παρακάτω παραδείγματα.

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Πίνακας προϊόντων - Παραδείγματα των συναρτήσεων Count, Avg, Sum

(Πηγή: https://www.w3schools.com/sql/sql_count_avg_sum.asp)

Παράδειγμα συνάρτησης COUNT()

Εκτελέστε ένα ερώτημα για να βρείτε τον αριθμό των προϊόντων:

```
SELECT COUNT(ΚωδικόςΠροϊόντων)
FROM Προϊόντα;
```

Παράδειγμα συνάρτησης AVG()

Εκτελέστε ένα ερώτημα για να βρείτε τη μέση τιμή όλων των προϊόντων:

```
SELECT AVG(Τιμή)
FROM Προϊόντα;
```

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

Πίνακας με πληροφορίες παραγγελιών - Παραδείγματα των συναρτήσεων Count, Avg, Sum

(Πηγή: https://www.w3schools.com/sql/sql_count_avg_sum.asp)

Στο παρακάτω παράδειγμα, θα χρησιμοποιήσουμε τον πίνακα ΛεπτομέρειεςΠαραγγελιών, που φαίνεται παραπάνω, για να βρούμε το άθροισμα της «Ποσότητας»:

```
SELECT SUM(Ποσότητα)
FROM ΛεπτομέρειεςΠαραγγελιών;
```

Τελεστής LIKE στην SQL

Ο τελεστής LIKE χρησιμοποιείται σε ένα όρο WHERE για να αναζητήσει ένα καθορισμένο μοτίβο σε μια στήλη.

Σύνταξη του τελεστή LIKE

```
SELECT στήλη1, στήλη2, ...
FROM όνομα_πίνακα
WHERE στήλη LIKE μοτίβο;
```

Ακολουθούν μερικά παραδείγματα που δείχνουν διαφορετικούς τελεστές LIKE με χαρακτήρες μπαλαντέρ '%' και '_':

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

(Πηγή: https://www.w3schools.com/sql/sql_like.asp)

Θα δούμε μερικά παραδείγματα που θα χρησιμοποιούν τις ελληνικές αντιστοιχίες από τον παρακάτω πίνακα Πελατών.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Πίνακας πελατών - παραδείγματα του τελεστή LIKE

(Πηγή: https://www.w3schools.com/sql/sql_like.asp)

Παράδειγμα 1: Επιλογή όλων των πελατών με Όνομα Πελάτη που αρχίζει με ‘α’:

```
SELECT * FROM Πελάτες
WHERE ΌνομαΠελάτη LIKE 'α%';
```

Παράδειγμα 2: Επιλογή όλων των ονομάτων πελατών που τελειώνουν με «α»:

```
SELECT * FROM Πελάτες
WHERE ΌνομαΠελάτη LIKE '%α';
```

Παράδειγμα 3: Επιλογή όλων των ονομάτων πελατών που περιέχουν ‘ή’ στο όνομά τους σε οποιαδήποτε θέση.

```
SELECT * FROM Πελάτες
WHERE Όνομα Πελάτη LIKE '%ή%';
```

Παράδειγμα 4: Επιλογή όλων των ονομάτων πελατών που περιέχουν ‘ρ’ στη δεύτερη θέση:

```
SELECT * FROM Πελάτες
WHERE ΌνομαΠελάτη LIKE '_ρ%';
```

Παράδειγμα 5: Επιλογή όλων των ονομάτων πελατών που αρχίζουν με ‘α’ και έχουν τουλάχιστον τρεις χαρακτήρες στο σύνολο

```
SELECT * FROM Πελάτες
WHERE ΌνομαΠελάτη LIKE 'α__%';
```

Παράδειγμα 6: Επιλογή όλων των ονομάτων πελατών που αρχίζουν με ‘α’ και τελειώνουν με ‘ο’ :

```
SELECT * FROM Πελάτες
WHERE ΌνομαΠελάτη LIKE 'α%ο';
```

Παράδειγμα 7: Επιλογή όλων των ονομάτων πελατών που δεν αρχίζουν με ‘α’.

```
SELECT * FROM Πελάτες
WHERE ΌνομαΠελάτη LIKE 'α%';
```

Χαρακτήρες Μπαλαντέρ στην SQL

Ένας χαρακτήρας μπαλαντέρ αντικαθιστά έναν ή περισσότερους χαρακτήρες σε μια συμβολοσειρά. Χρησιμοποιείται με τον τελεστή LIKE.

Ο τελεστής LIKE χρησιμοποιείται επίσης σε έναν όρο WHERE για να αναζητήσει ένα συγκεκριμένο μοτίβο σε μια στήλη, όπως έχουμε δει στην προηγούμενη υποενότητα.

Χαρακτήρες Μπαλαντέρ στο MS Access

Symbol	Description	Example
*	Represents zero or more characters	bl* finds bl, black, blue, and blob
?	Represents a single character	h?t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
!	Represents any character not in the brackets	h[!oa]t finds hit, but not hot and hat
-	Represents any single character within the specified range	c[a-b]t finds cat and cbt
#	Represents any single numeric character	2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295

(Πηγή: https://www.w3schools.com/sql/sql_wildcards.asp)

Χαρακτήρες Μπαλαντέρ στον διακομιστή SQL

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents any single character within the specified range	c[a-b]t finds cat and cbt

(Πηγή: https://www.w3schools.com/sql/sql_wildcards.asp)

Οι χαρακτήρες μπαλαντέρ μπορούν να χρησιμοποιηθούν συνδυαστικά. Δείτε μερικά παραδείγματα στον παρακάτω πίνακα:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a__%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

Παραδείγματα χαρακτήρων μπαλαντέρ με % και '_'
(Πηγή: https://www.w3schools.com/sql/sql_wildcards.asp)

Ας δούμε μερικά παραδείγματα χρησιμοποιώντας τον πίνακα Πελατών.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK

Πίνακας πελατών - παράδειγμα χαρακτήρων ΜΠΑΛΑΝΤΕΡ

(Πηγή: https://www.w3schools.com/sql/sql_wildcards.asp)

Παραδείγματα με τον χαρακτήρα μπαλαντέρ %

Σε αυτό το παράδειγμα, επιλέγουμε όλους τους πελάτες με Πόλη που ξεκινά με «βερ»:

```
SELECT * FROM Πελάτες
WHERE Πόλη LIKE ‘βερ%';
```

Στο παρακάτω παράδειγμα, επιλέγουμε όλους τους πελάτες με Πόλη που περιέχει το «εσ»:

```
SELECT * FROM Πελάτες
WHERE Πόλη LIKE ‘%εσ%';
```

Παραδείγματα με τον χαρακτήρα μπαλαντέρ «_»

Εδώ, επιλέγουμε όλους τους πελάτες που μένουν σε Πόλη ξεκινά με οποιονδήποτε χαρακτήρα ακολουθούμενο από «ονδίνο»:

```
SELECT * FROM Πελάτες
WHERE Πόλη LIKE ‘_ονδίνο';
```

Σε αυτό το παράδειγμα, επιλέγουμε ξανά όλους τους πελάτες από Πόλη που ξεκινά με ‘Λ’, ακολουθούμενο από οποιονδήποτε χαρακτήρα, ακολουθούμενο από «νδ», ακολουθούμενο από οποιονδήποτε χαρακτήρα, ακολουθούμενο από «νο»:

```
SELECT * FROM Πελάτες
```

WHERE Πόλη LIKE ‘Λ_νδ_νο';

Παραδείγματα με τον χαρακτήρα μπαλαντέρ [charlist]
Εδώ, επιλέγουμε όλους τους πελάτες από Πόλη που ξεκινά από «β», «σ» ή «π»:
SELECT * FROM Πελάτες
WHERE Πόλη LIKE ‘[βσπ]%';

Στο παρακάτω παράδειγμα, επιλέγουμε όλους τους πελάτες από Πόλη που να ξεκινά από «α», «β» ή «γ»:
SELECT * FROM Πελάτες
WHERE Πόλη LIKE '[α-γ]%';

Παραδείγματα με τον χαρακτήρα μπαλαντέρ [!charlist]
Το θαυμαστικό εμφανίζει χαρακτήρες που δεν περιέχουν μια καθορισμένη συμβολοσειρά. Για παράδειγμα, θέλουμε να επιλέξουμε όλους τους πελάτες από Πόλη που δεν ξεκινά από «β», «σ» ή «π»:
SELECT * FROM Πελάτες
WHERE Πόλη LIKE ‘[!βσπ]%';

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε τα ακόλουθα:
SELECT * FROM Πελάτες
WHERE Πόλη NOT LIKE ‘[βσπ]%';

Τελεστής In στην SQL

Ο τελεστής IN χρησιμοποιείται για τον καθορισμό πολλαπλών τιμών σε έναν όρο WHERE. Μπορεί να θεωρηθεί ότι πληροί διάφορες προϋποθέσεις.

Σύνταξη 1:

SELECT όνομα(τα)_στήλης(ών)

FROM όνομα_πίνακα

WHERE όνομα_στήλης IN (τιμή1, τιμή2, ...);

Σύνταξη 2:

SELECT όνομα(τα)_στήλης(ών)

FROM όνομα_πίνακα

WHERE όνομα_στήλης IN (εντολή SELECT);

Υπάρχουν δύο τρόποι για να χρησιμοποιήσετε τον τελεστή IN, όπως έχουμε δει.

Ας υποθέσουμε ότι έχουμε έναν πίνακα που ονομάζεται «Πελάτες» που περιέχει τις ακόλουθες στήλες: ΚωδικόςΠελάτη, ΌνομαΠελάτη, ΌνομαΕπικοινωνίας, Διεύθυνση, Πόλη, Ταχυδρομικός Κώδικας και Χώρα.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain

Πίνακας Πελατών - Παράδειγμα τελεστή IN

(Πηγή: https://www.w3schools.com/sql/sql_in.asp)

Για παράδειγμα, θέλουμε να επιλέξουμε όλους τους πελάτες που βρίσκονται στη Γερμανία, τη Γαλλία ή το Ηνωμένο Βασίλειο:

SELECT * FROM Πελάτες

WHERE Χώρα IN ('Γερμανία', 'Γαλλία', 'Ηνωμένο Βασίλειο')

Ένα άλλο παράδειγμα είναι η επιλογή όλων των πελατών που **δεν** βρίσκονται στη Γερμανία, τη Γαλλία ή το Ηνωμένο Βασίλειο:

```
SELECT * FROM Πελάτες
WHERE Χώρα NOT IN (‘Γερμανία’, ‘Γαλλία’, ‘Ηνωμένο Βασίλειο’)
```

Ας εξετάσουμε επίσης ένα τρίτο παράδειγμα όπου θέλουμε να επιλέξουμε πελάτες που προέρχονται από τις ίδιες χώρες με τους προμηθευτές:

```
SELECT * FROM Πελάτες
WHERE Χώρα IN (SELECT Χώρα FROM Προμηθευτές)
```

Τελεστής Between στην SQL

Ο τελεστής BETWEEN παρέχει ένα εύρος τιμών από τις οποίες μπορείτε να επιλέξετε. Οι τιμές μπορεί να είναι κείμενο, αριθμοί ή ημερομηνίες. Ο τελεστής BETWEEN περιλαμβάνει τις τιμές έναρξης και λήξης.

Σύνταξη:

```
SELECT όνομα(τα)_στήλης(ών)
FROM όνομα_πίνακα
WHERE όνομα_στήλης BETWEEN τιμή1 AND τιμή2;
```

Ας υποθέσουμε ότι έχουμε τον ακόλουθο πίνακα, ο οποίος περιέχει πληροφορίες για διαφορετικά προϊόντα.

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	1	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	1	2	36 boxes	21.35

Πίνακας - Παράδειγμα τελεστή BETWEEN

(Πηγή: https://www.w3schools.com/sql/sql_between.asp)

Θα υπάρχει μια σειρά παραδειγμάτων με τους ακόλουθους τελεστές: BETWEEN, NOT BETWEEN, BETWEEN με IN, BETWEEN και NOT BETWEEN με τιμές κειμένου και ημερομηνίες.

Παράδειγμα BETWEEN: Εμφάνιση όλων των προϊόντων με εύρος τιμών 10 έως 20.

```
SELECT * FROM Πρόϊόντα
WHERE Τιμή BETWEEN 10 AND 20;
```

Παράδειγμα NOT BETWEEN: Εμφάνιση όλων των προϊόντων εκτός της σειράς που ορίσαμε στο προηγούμενο παράδειγμα.

```
SELECT * FROM Προϊόντα
WHERE Τιμή NOT BETWEEN 10 AND 20;
```

Παράδειγμα BETWEEN με IN: Εμφάνιση όλων των προϊόντων με εύρος τιμών 10 έως 20 και δεν εμφανίζει προϊόντα με αναγνωριστικό κατηγορίας 1, 2 ή 3.

```
SELECT * FROM Προϊόντα
WHERE Τιμή BETWEEN 10 AND 20
AND ΑριθμόςΚατηγορίας NOT IN (1,2,3);
```

Παράδειγμα BETWEEN με τιμές κειμένου: Εμφάνιση όλων των πρόϊόντων με Όνομα προϊόντος μεταξύ Carnarvon Tigers και Mozzarella di Giovanni.

```
SELECT * FROM Προϊόντα
WHERE ΌνομαΠροϊόντος BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ΌνομαΠροϊόντος;
```

Παράδειγμα NOT BETWEEN με τιμές κειμένου: Εμφάνιση όλων των προϊόντων με Όνομα προϊόντος που δεν είναι μεταξύ Carnarvon Tigers και Mozzarella di Giovanni.

```
SELECT * FROM Προϊόντα
WHERE ΌνομαΠροϊόντος NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ΌνομαΠροϊόντος;
```

Υποθέστε ότι έχουμε τον ακόλουθο πίνακα που περιέχει πληροφορίες για διαφορετικές παραγγελίες:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/9/1996	1
10252	76	4	7/10/1996	2

Πίνακας - Παράδειγμα τελεστή BETWEEN

(Πηγή: https://www.w3schools.com/sql/sql_between.asp)

Παράδειγμα BETWEEN σε ημερομηνίες: Εμφάνιση όλων των παραγγελιών με ημερομηνία παραγγελίας μεταξύ '01-Ιουλίου-1996' και '31-Ιουλίου-1996'

Υπάρχουν δύο τρόποι για να γίνει αυτό, χρησιμοποιώντας είτε ένα hashtag (#) ή εισαγωγικά (‘’):

```
SELECT * FROM Παραγγελίες
WHERE ΗμερομηνίαΠαραγγελίας BETWEEN #07/01/1996# AND #07/31/1996#;
```

Ή

```
SELECT * FROM Παραγγελίες
WHERE ΗμερομηνίαΠαραγγελίας BETWEEN '1996-07-01' AND '1996-07-31';
```

Ψευδώνυμα στην SQL (Aliases)

Τα ψευδώνυμα δίνουν ένα προσωρινό όνομα σε έναν πίνακα ή μια στήλη μέσα σε έναν πίνακα. Ένα ψευδώνυμο υπάρχει μόνο για τη διάρκεια ενός ερωτήματος και χρησιμοποιείται συνήθως για να κάνει τα ονόματα των στηλών πιο ευανάγνωστα. Ένα ψευδώνυμο δημιουργείται χρησιμοποιώντας τη λέξη-κλειδί **AS.**

Σύνταξη για ψευδώνυμο στήλης:

```
SELECT όνομα_στήλης AS όνομα_ψευδώνυμου
FROM όνομα_πίνακα;
```

Σύνταξη για το ψευδώνυμο πίνακα:

```
SELECT όνομα(τα)_στήλης(ών)
FROM όνομα_πίνακα AS όνομα_ψευδώνυμου;
```

Ψευδώνυμα στηλών

Ας δούμε ένα παράδειγμα που δημιουργεί δύο ψευδώνυμα, ένα για κάθε στήλη:

```
SELECT ΚωδικόςΠελάτη AS ID, ΌνομαΠελάτη AS Πελάτης
FROM Πελάτες;
```

Ένα άλλο παράδειγμα δημιουργεί και πάλι δύο ψευδώνυμα:

```
SELECT ΌνομαΠελάτη AS Πελάτης, ΌνομαΕπικοινωνίας AS [Άτομο Επικοινωνίας]
FROM Πελάτες;
```

Σημειώστε ότι τοποθετείται σε αγκύλες ([ ]) επειδή το ψευδώνυμο περιέχει κενά. Τα εισαγωγικά μπορούν να χρησιμοποιηθούν ως εναλλακτική λύση για τις αγκύλες.

Έχετε επίσης την επιλογή να δημιουργήσετε ένα ψευδώνυμο που περιέχει μία ή περισσότερες στήλες, ας δούμε το παρακάτω παράδειγμα για να δούμε πώς λειτουργεί:

```
SELECT ΌνομαΠελάτη, Διεύθυνση + ', ' + ΤαχυδρομικόςΚώδικας + ' ' + Πόλη + ', ' + Χώρα AS Διεύθυνση
FROM Πελάτες;
```

Η παραπάνω εντολή αλλάζει λίγο στο MySQL:

```
SELECT ΌνομαΠελάτη, CONCAT(Διεύθυνση,', ',ΤαχυδρομικόςΚώδικας,', ',Πόλη,', ',Χώρα) AS Διεύθυνση
FROM Πελάτες;
```

Ψευδώνυμα πίνακα

Το παρακάτω παράδειγμα επιλέγει όλες τις παραγγελίες από τον πίνακα πελατών με ΚωδικόΠελάτη =4 δίνοντας το ψευδώνυμο “Around the Horn”.

Εδώ χρησιμοποιούνται ψευδώνυμα για τη συντόμευση του ερωτήματος:

```
SELECT o.ΚωδικόςΠαραγγελίας, o.ΗμερομηνίαΠαραγγελίας, c.ΌνομαΠελάτη
FROM Πελάτες AS c, Παραγγελίες AS o
WHERE c.Όνομα Πελάτη='Around the Horn' AND
c.ΚωδικόςΠελάτη=o.ΚωδικόςΠελάτη;
```

Ένα ερώτημα χωρίς ψευδώνυμα θα έμοιαζε κάπως έτσι:

```
SELECT Παραγγελίες.ΚωδικόςΠαραγγελίας, Παραγγελίες.ΗμερομηνίαΠαραγγελίας, Πελάτες.ΌνομαΠελάτη
FROM Πελάτες, Παραγγελίες
WHERE Πελάτες.ΌνομαΠελάτη='Around the Horn' AND
Πελάτες.ΚωδικόςΠελάτη=Παραγγελίες.ΚωδικόςΠελάτη;
```

Συνενώσεις στην SQL (JOIN)

Μια εντολή JOIN συνδυάζει σειρές από δύο ή περισσότερους πίνακες με βάση μια σχετική στήλη που βρίσκεται και στους δύο πίνακες.

Ας δούμε τον πίνακα Παραγγελίες και τον πίνακα Πελάτες:

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Πίνακες Παραγγελιών και Πελατών - Παράδειγμα συνένωσης JOIN
(Πηγή: https://www.w3schools.com/sql/sql_join.asp)

Αν κοιτάξετε τους δύο πίνακες, θα παρατηρήσετε μια κοινή στήλη που ονομάζεται ΚωδικόςΠελάτη (CustomerID).
Με βάση την κοινή στήλη, μπορούμε να δημιουργήσουμε μια εντολή SQL που χρησιμοποιεί το INNER JOIN, η οποία επιλέγει εγγραφές που έχουν τις ίδιες τιμές και στους δύο πίνακες.

```
SELECT Παραγγελίες.ΚωδικόςΠαραγγελίας, Πελάτες.ΌνομαΠελάτη,
Παραγγελίες.ΗμερομηνίαΠαραγγελίας
FROM Παραγγελίες
INNER JOIN Πελάτες ON Παραγγελίες.ΚωδικόςΠελάτη = Πελάτες.ΚωδικόςΠελάτη;
```

Αυτή η εντολή θα δημιουργήσει κάτι σαν τον ακόλουθο πίνακα:

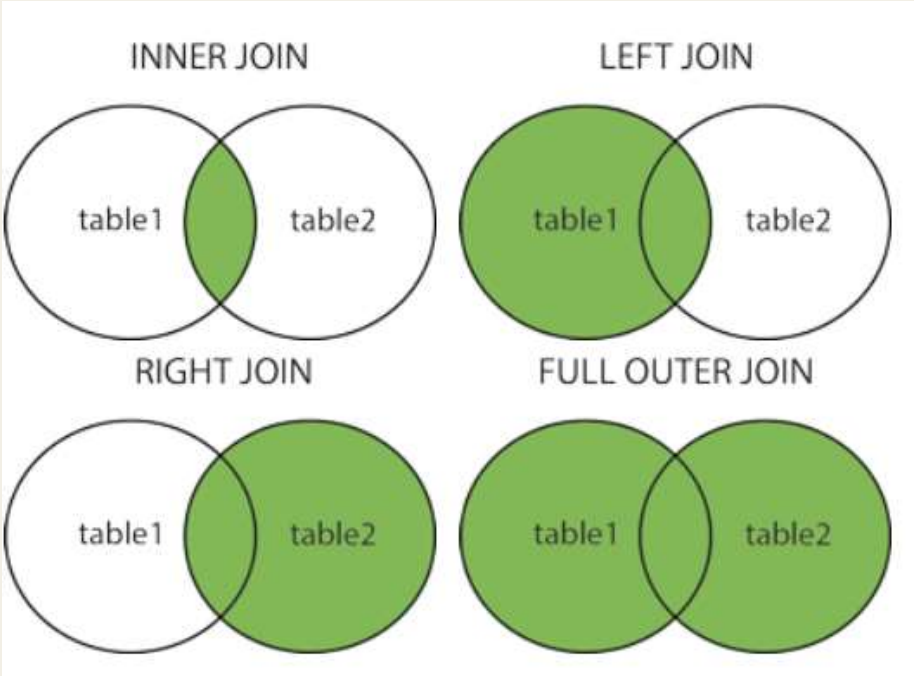
OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Πίνακας - παράδειγμα της συνένωσης JOIN

(Πηγή: https://www.w3schools.com/sql/sql_join.asp)

Υπάρχουν τέσσερις διαφορετικές εντολές JOIN στην SQL:

1. (INNER) JOIN: Επιλέγει εγγραφές που έχουν αντίστοιχες τιμές και στους δύο πίνακες.
2. LEFT (OUTER) JOIN: Επιλέγει όλες τις εγγραφές από τον πίνακα στα αριστερά και τις αντίστοιχες αντιστοιχισμένες εγγραφές από τον πίνακα στα δεξιά.
3. RIGHT (OUTER) JOIN: Επιλέγει όλες τις εγγραφές από τον πίνακα στα δεξιά και τις αντίστοιχες αντιστοιχισμένες εγγραφές από τον πίνακα στα αριστερά.
4. FULL (OUTER) JOIN: Επιλέγει όλες τις εγγραφές όταν υπάρχει αντιστοιχία είτε στον πίνακα στα αριστερά ή στα δεξιά.



Εικόνα - Διαφορετικοί τύποι εντολών JOIN

(Πηγή: https://www.w3schools.com/sql/sql_join.asp)

Συνένωση Inner join στην SQL

Η λέξη-κλειδί INNER JOIN επιλέγει εγγραφές που έχουν ίδιες τιμές και από τους δύο πίνακες.

Σύνταξη:

SELECT όνομα(τα)_στήλης(ών)

FROM πίνακας1

INNER JOIN πίνακας2

ON πίνακας1.όνομα_στήλης = πίνακας2.όνομα_στήλης;

Οι ίδιοι πίνακες με το παράδειγμα της προηγούμενης υποενότητας χρησιμοποιούνται για την εκτέλεση μιας εσωτερικής ένωσης.

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujilio Emparedados y helados	Ana Trujilio	Mexico
3	Antonio Moreno Taqueria	Antonio Moreno	Mexico

Πίνακες Παραγγελιών και Πελατών - Παράδειγμα συνένωσης JOIN
(Πηγή: https://www.w3schools.com/sql/sql_join_inner.asp)

Σε αυτό το παράδειγμα, θέλουμε να ανακτήσουμε τα ονόματα των πελατών και τους αντίστοιχους κωδικούς της παραγγελίας τους:

SELECT Παραγγελίες.ΚωδικόςΠαραγγελίας, Πελάτες.ΌνομαΠελάτη
FROM Παραγγελίες
INNER JOIN Πελάτες ON Παραγγελίες.ΚωδικόςΠελάτη = Πελάτες.ΚωδικόςΠελάτη;

Σημειώστε ότι η λέξη κλειδί INNER JOIN θα επιλέξει όλες τις σειρές και από τους δύο πίνακες που έχουν τις ίδιες τιμές. Εάν οι εγγραφές στον πίνακα Παραγγελίες δεν έχουν αντιστοιχίσεις στον πίνακα Πελάτες, δεν θα επιλεγούν.

Στο παρακάτω παράδειγμα, θα δούμε πώς μπορείτε να ενώσετε τρεις πίνακες που περιέχουν πληροφορίες πελατών και αποστολέων:

```
SELECT Παραγγελίες.ΚωδικόςΠαραγγελίας, Πελάτες.ΌνομαΠελάτη,
Αποστολέας.ΌνομαΑποστολέα
FROM ((Παραγγελίες
INNER JOIN Πελάτες ON Παραγγελίες.ΚωδικόςΠελάτη = Πελάτες.ΚωδικόςΠελάτη)
INNER JOIN Αποστολείς ON Παραγγελίες.ΚωδικόςΑποστολέα =
Αποστολείς.ΚωδικόςΑποστολέα);
```

Συνένωση Left Join στην SQL

Η λέξη-κλειδί LEFT JOIN επιλέγει όλες τις εγγραφές από τον πίνακα στα αριστερά και τις αντίστοιχες εγγραφές από τον πίνακα στα δεξιά. Εάν δεν βρεθούν αντιστοιχίσεις, θα εμφανιστούν μηδενικές εγγραφές από τον πίνακα στα δεξιά.

Σύνταξη:

```
SELECT όνομα(τα)_στήλης(ών)
FROM πίνακας1
LEFT JOIN πίνακας2
ON πίνακας1.όνομα_στήλης = πίνακας2.όνομα_στήλης;
```

Σημειώστε ότι η εντολή LEFT JOIN ονομάζεται LEFT OUTER JOIN σε ορισμένες βάσεις δεδομένων.

Για παράδειγμα, ας επιλέξουμε όλους τους πελάτες και τυχόν παραγγελίες που μπορεί να έχουν αυτοί οι πελάτες:

```
SELECT Πελάτες.ΌνομαΠελάτη, Παραγγελίες.ΚωδικόςΠαραγγελίας
FROM Πελάτες
```

LEFT JOIN Παραγγελίες ON Πελάτες.ΚωδικόςΠελάτη = Παραγγελίες.ΚωδικόςΠελάτη
ORDER BY Πελάτες.ΌνομαΠελάτη;

Σημειώστε ότι θα εμφανιστούν όλες οι εγγραφές από τον αριστερό πίνακα με όνομα Πελάτες, ακόμη και αν δεν υπάρχουν αντιστοιχίσεις στον δεξιό πίνακα με όνομα Παραγγελίες.

Συνένωση Right join στην SQL

Η λέξη-κλειδί RIGHT JOIN ακολουθεί ουσιαστικά την ίδια λογική ξεκινώντας από τη δεξιά πλευρά αντί για την αριστερή όπως περιγράφεται στην προηγούμενη υποενότητα.

Η εντολή RIGHT JOIN θα εμφανίσει όλες τις εγγραφές από τον πίνακα στα δεξιά και τις αντίστοιχες εγγραφές από τον πίνακα στα αριστερά, εάν υπάρχουν.

Σύνταξη:

SELECT όνομα(τα)_στήλης(ών)
FROM πίνακας1
RIGHT JOIN πίνακας2
ON πίνακας1.όνομα_στήλης = πίνακας2.όνομα_στήλης;

Εξετάστε τους ακόλουθους δύο πίνακες, τους πίνακες Παραγγελιών και Εργαζομένων:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Πίνακας Παραγγελιών - παράδειγμα συνένωσης RIGHT JOIN
(Πηγή: https://www.w3schools.com/sql/sql_join_right.asp)

EmployeeID	LastName	FirstName	BirthDate	Photo
1	Davolio	Nancy	12/8/1968	EmpID1.pic
2	Fuller	Andrew	2/19/1952	EmpID2.pic
3	Leverling	Janet	8/30/1963	EmpID3.pic

Πίνακας Εργαζομένων - παράδειγμα συνένωσης RIGHT JOIN

(Πηγή: https://www.w3schools.com/sql/sql_join_right.asp)

Το ακόλουθο παράδειγμα θα εμφανίσει όλους τους εργαζομένους και τυχόν παραγγελίες που μπορεί να έχουν κάνει:

```
SELECT Παραγγελίες.ΚωδικόςΠαραγγελίας, Εργαζόμενοι.Επίθετο,
Εργαζόμενοι.Όνομα
FROM Παραγγελίες
RIGHT JOIN Εργαζόμενοι ON Παραγγελίες.ΚωδικόςΕργαζομένου =
Εργαζόμενοι.ΚωδικόςΕργαζομένου
ORDER BY Παραγγελίες.ΚωδικόςΠαραγγελίας;
```

Σημειώστε ότι η λέξη κλειδί RIGHT JOIN εμφανίζει όλα τα αρχεία από τον πίνακα στα δεξιά με όνομα Εργαζόμενοι, ακόμα και αν δεν βρεθούν αντιστοιχίες στον αριστερό πίνακα με όνομα Παραγγελίες. Το ίδιο ισχύει και για την εντολή LEFT JOIN που είδαμε νωρίτερα.

Συνένωση Full Join στην SQL

Η λέξη-κλειδί FULL JOIN επιλέγει όλες τις εγγραφές όταν οι εγγραφές που είναι ίδιες βρίσκονται είτε στον πίνακα στα δεξιά είτε στα αριστερά.

Σημειώστε ότι οι εντολές FULL OUTER JOIN και FULL JOIN είναι το ίδιο πράγμα.

Σύνταξη:

```
SELECT όνομα(τα)_στήλης(ών)
FROM πίνακας1
FULL OUTER JOIN πίνακας2
```

ON πίνακας1.όνομα_στήλης = πίνακας2.όνομα_στήλης
WHERE συνθήκη;

Λάβετε υπόψη ότι μια εντολή FULL JOIN μπορεί δυνητικά να επιστρέψει μεγάλα σύνολα αποτελεσμάτων.

Παρατηρήστε τους ακόλουθους δύο πίνακες, τον πίνακα Παραγγελιών και τον πίνακα Πελατών:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Πίνακας Παραγγελιών - παράδειγμα συνένωση FULL JOIN
(Πηγή: https://www.w3schools.com/sql/sql_join_full.asp)

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Πίνακας Πελατών - παράδειγμα συνένωση FULL JOIN
(Πηγή: https://www.w3schools.com/sql/sql_join_full.asp)

Ένα παράδειγμα που επιλέγει όλους τους πελάτες και τις παραγγελίες:
SELECT Πελάτες.ΌνομαΠελάτη, Παραγγελλίες.ΚωδικόςΠαραγγελίας

```
FROM Πελάτες
FULL OUTER JOIN Παραγγελίες ON Πελάτες.ΚωδικόςΠελάτη =Παραγγελίες.ΚωδικόςΠελάτη
ORDER BY Πελάτες.ΌνομαΠελάτη;
```

Το αποτέλεσμα της εντολής FULL JOIN μπορεί να μοιάζει κάπως έτσι:

CustomerName	OrderID
Null	10309
Null	10310
Alfreds Futterkiste	*Null*
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	*Null*

Εντολή FULL JOIN σε πίνακες Πελατών και Παραγγελιών - παράδειγμα συνένωσης FULL JOIN

(Πηγή: https://www.w3schools.com/sql/sql_join_full.asp)

Εδώ μπορούμε να δούμε ότι εμφανίζει όλες τις εγγραφές που ταιριάζουν και από τους δύο πίνακες, ακόμη και αν δεν υπάρχουν κοινές αντιστοιχίες μεταξύ των δύο πινάκων. Σε περίπτωση που δεν υπάρχουν κοινές αντιστοιχίσεις, ορίζεται μηδενική τιμή.

Συνένωση Self-Join στην SQL

Η συνένωση SELF-JOIN θεωρείται ως μια κανονική συνένωση JOIN, ωστόσο οι εγγραφές στον πίνακα συσχετίζονται αυτόματα.

Σύνταξη:

```
SELECT όνομα(τα)_στήλης(ών)
FROM πίνακας1 T1, πίνακας1 T2
WHERE συνθήκη;
```

Τα T1 και T2 είναι ψευδώνυμα που χρησιμοποιούνται για τον ίδιο πίνακα.

Ας πάρουμε τον πίνακα Πελατών ως παράδειγμα:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Πίνακας Πελατών - παράδειγμα συνένωσης SELF JOIN
(Πηγή: https://www.w3schools.com/sql/sql_join_self.asp)

Εδώ, θέλουμε να επιλέξουμε πελάτες που προέρχονται από την ίδια πόλη:

SELECT A.ΌνομαΠελάτη AS ΌνομαΠελάτη1, B.ΌνομαΠελάτη AS ΌνομαΠελάτη2, A.Πόλη

FROM Πελάτες A, Πελάτες B

WHERE A.ΚωδικόςΠελάτη <> B.ΚωδικόςΠελάτη

AND A.Πόλη = B.Πόλη

ORDER BY A.Πόλη;

Ένωση στην SQL (UNION)

Ο τελεστής UNION χρησιμοποιείται για να συνδυάσει το σύνολο δύο ή περισσότερων εντολών SELECT.

Υπάρχουν ορισμένες απαιτήσεις για να καταστεί δυνατός ένας τελεστής UNION:

1. Κάθε εντολή SELECT εντός του τελεστή UNION πρέπει να έχει τον ίδιο αριθμό στηλών

2. Οι στήλες πρέπει να έχουν παρόμοιους τύπους δεδομένων.

3. Οι στήλες σε κάθε εντολή SELECT πρέπει να είναι στην ίδια σειρά.

Σύνταξη:

```
SELECT όνομα_στήλης(ών) FROM πίνακας1
UNION
SELECT όνομα_στήλης(ών) FROM πίνακας2;
```

Σημειώστε ότι η λειτουργία UNION επιλέγει μόνο διακριτές τιμές από προεπιλογή.

Για να επιτρέψετε τις διπλές τιμές, χρησιμοποιήστε την λειτουργία UNION ALL:

```
SELECT όνομα(τα)_στήλης(ών) FROM πίνακας1
UNION ALL
SELECT όνομα(τα)_στήλης(ών) FROM πίνακας2;
```

Σημειώστε ότι τα ονόματα των στηλών στις δύο εντολές SELECT είναι συνήθως ίσα.

Τώρα ας δούμε μερικά παραδείγματα της ένωσης UNION, της UNION ALL, και UNION με εντολές που χρησιμοποιούν τον όρο WHERE για να καταλάβουμε λίγο καλύτερα πώς μπορούμε να το χρησιμοποιήσουμε.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Πίνακας Πελατών - Παράδειγμα τελεστή UNION

(Πηγή: https://www.w3schools.com/sql/sql_union.asp)

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

Πίνακας προμηθευτών - Παράδειγμα τελεστή UNION

(Πηγή: https://www.w3schools.com/sql/sql_union.asp)

Το πρώτο παράδειγμα χρησιμοποιείται για την επιλογή ξεχωριστών πόλεων από τους δύο πίνακες που παρουσιάζονται παραπάνω:

```
SELECT Πόλη FROM Πελάτες
UNION
SELECT Πόλη FROM Προμηθευτές
ORDER BY Πόλη;
```

Εφόσον χρησιμοποιούμε τον τελεστή UNION, οι προμηθευτές από την ίδια πόλη θα αναφέρονται μόνο μία φορά. Αν θέλετε να δείτε τις διπλές τιμές, χρησιμοποιήστε τον τελεστή UNION ALL.

Το ακόλουθο παράδειγμα θα κάνει ακριβώς αυτό και θα εμφανίσει τυχόν διπλές τιμές και από τους δύο πίνακες:

```
SELECT Πόλη FROM Πελάτες
UNION ALL
SELECT Πόλη FROM Προμηθευτές
ORDER BY Πόλη;
```

Το ακόλουθο παράδειγμα θα εμφανίσει τις ξεχωριστές γερμανικές πόλεις από τους πίνακες «Πελάτες» και «Προμηθευτές» με τη χρήση του όρου WHERE:

```
SELECT Πόλη, Χώρα FROM Πελάτες

WHERE Χώρα='Γερμανία'

UNION
```

SELECT Πόλη, Χώρα FROM Πελάτες

WHERE Χώρα = ‘Γερμανία’

ORDER BY Πόλη;

Αυτό το παράδειγμα είναι παρόμοιο με το προηγούμενο, ωστόσο εδώ εμφανίζονται πιθανές διπλές τιμές:

```
SELECT Πόλη, Χώρα FROM Πελάτες
WHERE Χώρα = ‘Γερμανία’
UNION ALL
SELECT Πόλη, Χώρα FROM Πελάτες
WHERE Χώρα = ‘Γερμανία’
ORDER BY Πόλη;
```

Ένα άλλο παράδειγμα θα απαριθμήσει όλους τους πελάτες και τους προμηθευτές:

```
SELECT Πελάτης AS Είδος, ΌνομαΕπικοινωνίας, Πόλη, Χώρα
FROM Πελάτες
UNION
SELECT 'Προμηθευτής', ΌνομαΕπικοινωνίας, Πόλη, Χώρα
FROM Προμηθευτές;
```

Όπως βλέπετε, εδώ χρησιμοποιήσαμε τον όρο AS για να δημιουργήσουμε ένα ψευδώνυμο για το δεδομένο ερώτημα που θα εξαφανιστεί μετά την ολοκλήρωσή του.

Ομαδοποίηση στην SQL (Group By)

Η εντολή GROUP BY ομαδοποιεί τις σειρές με τις ίδιες τιμές σε συνοπτικές γραμμές. Για παράδειγμα, σκεφτείτε ότι θέλετε να βρείτε τον αριθμό των πελατών σε κάθε χώρα.

Επίσης, η εντολή GROUP BY χρησιμοποιείται συχνά με αθροιστικές συναρτήσεις όπως COUNT(), MAX(), MIN(), SUM(), AVG() για την ομαδοποίηση του αποτελέσματος κατά μία ή περισσότερες στήλες.

Σύνταξη:

SELECT όνομα_στήλης(ών)
FROM όνομα_πίνακα
WHERE συνθήκη
GROUP BY όνομα_στήλης(ών)
ORDER BY όνομα_στήλης(ών);

Θα χρησιμοποιήσουμε τον πίνακα Πελατών, που φαίνεται παρακάτω, στα παραδείγματα μας.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Πίνακας Πελατών - παράδειγμα εντολής GROUP BY
(Πηγή: https://www.w3schools.com/sql/sql_groupby.asp)

Ως πρώτο παράδειγμα, θα απαριθμήσουμε τον αριθμό των πελατών που βρέθηκαν σε κάθε χώρα:

SELECT COUNT(ΚωδικόςΠελάτη), Χώρα
FROM Πελάτες
GROUP BY Χώρα;

Ως δεύτερο παράδειγμα, θα απαριθμήσουμε και πάλι τον αριθμό των πελατών σε κάθε χώρα, αλλά με φθίνουσα σειρά.

```
SELECT COUNT(ΚωδικόςΠελάτη), Χώρα
FROM Πελάτες
GROUP BY Χώρα;
ORDER BY COUNT(ΚωδικόςΠελάτη) DESC;
```

Στο παρακάτω παράδειγμα, θα χρησιμοποιήσουμε την εντολή GROUP BY με τον τελεστή JOIN χρησιμοποιώντας τους πίνακες Παραγγελιών και Αποστολέων.

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2

Πίνακας Παραγγελιών - Παράδειγμα Group By

(Πηγή: https://www.w3schools.com/sql/sql_groupby.asp)

ShipperID	ShipperName
1	Speedy Express
2	United Package
3	Federal Shipping

Πίνακας αποστολέων - παράδειγμα εντολής GROUP BY

(Πηγή: https://www.w3schools.com/sql/sql_groupby.asp)

Σε αυτό το παράδειγμα, θα αναφέρουμε τον αριθμό των παραγγελιών που αποστέλλονται από κάθε αποστολέα:

```
SELECT Αποστολέας.ΌνομαΑποστολέα,
COUNT(Παραγγελίες.ΚωδικόςΠαραγγελίας) AS ΑριθμόςΠαραγγελιών
FROM Παραγγελίες
LEFT JOIN Αποστολείς ON Παραγγελίες.ΚωδικόςΑποστολέα=
Αποστολείς.ΚωδικόςΑποστολέα
GROUP BY ΌνομαΑποστολέα;
```

Όπως φαίνεται στο παραπάνω παράδειγμα, ξεκινήσαμε επιλέγοντας τα ονόματα των Αποστολέων από τον πίνακα Αποστολείς και απαριθμήσαμε τις παραγγελίες με βάση τον κωδικό παραγγελίας τους που αποθηκεύτηκε ως ψευδώνυμο.
Στη συνέχεια, χρησιμοποιήσαμε τον τελεστή LEFT JOIN για να συνδυάσουμε τον πίνακα Αποστολέων (πίνακας 2) και τον πίνακα Παραγγελιών (πίνακας 1) και να τους ομαδοποιήσουμε με βάση το όνομα του Αποστολέα.

Ο όρος HAVING στην SQL

Ο όρος HAVING προστέθηκε στην SQL επειδή ο όρος WHERE δεν μπορεί να χρησιμοποιηθεί με αθροιστικές συναρτήσεις.

Σύνταξη:

```
SELECT όνομα_στήλης(ών)
FROM όνομα_πίνακα
WHERE συνθήκη
GROUP BY όνομα_στήλης(ών)
HAVING συνθήκη
ORDER BY όνομα_στήλης(ών);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Πίνακας Πελατών - παράδειγμα του όρου HAVING
(Πηγή: https://www.w3schools.com/sql/sql_having.asp)

Σε αυτό το παράδειγμα, θα χρησιμοποιήσουμε ξανά τον πίνακα Πελατών. Εδώ, θέλουμε να απαριθμήσουμε τον αριθμό των πελατών που βρίσκονται σε κάθε χώρα, αλλά θέλουμε επίσης να συμπεριλάβουμε χώρες που έχουν περισσότερους από πέντε πελάτες.

```
SELECT COUNT(ΚωδικόςΠελάτη), Χώρα
FROM Πελάτες
GROUP BY Χώρα
HAVING COUNT(ΚωδικόςΠελάτη) > 5;
```

Σε αυτό το παράδειγμα, θέλουμε να απαριθμήσουμε ξανά τον αριθμό των πελατών ανά χώρα και να συμπεριλάβουμε χώρες με περισσότερους από πέντε πελάτες σε φθίνουσα σειρά.

```
SELECT COUNT(ΚωδικόςΠελάτη), Χώρα
FROM Πελάτες
GROUP BY Χώρα
HAVING COUNT(ΚωδικόςΠελάτη) > 5;
ORDER BY COUNT(ΚωδικόςΠελάτη) DESC;
```

Ας δοκιμάσουμε μερικά άλλα παραδείγματα συνδυάζοντας αυτά που έχουμε μάθει μέχρι στιγμής. Θα χρησιμοποιήσουμε τους πίνακες Παραγγελιών και Εργαζομένων στα ακόλουθα δύο παραδείγματα.

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2

Πίνακας Παραγγελιών - παράδειγμα HAVING

(Πηγή: https://www.w3schools.com/sql/sql_having.asp)

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	1968-12-08	EmpID1.pic	Education includes a BA....
2	Fuller	Andrew	1952-02-19	EmpID2.pic	Andrew received his BTS....
3	Leverling	Janet	1963-08-30	EmpID3.pic	Janet has a BS degree....

Πίνακας Παραγγελιών - παράδειγμα HAVING

(Πηγή: https://www.w3schools.com/sql/sql_having.asp)

Το ακόλουθο παράδειγμα θα απαριθμήσει τους εργαζομένους που έχουν καταχωρίσει περισσότερες από δέκα παραγγελίες:

```
SELECT Εργαζόμενοι.Επίθετο, COUNT(Παραγγελίες.ΚωδικόςΠαραγγελιών) AS ΑριθμόςΠαραγγελιών
FROM (Παραγγελίες
INNER JOIN Εργαζόμενοι ON Παραγγελίες.ΚωδικόςΕργαζομένου = Εργαζόμενοι.ΚωδικόςΕργαζομένου)
GROUP BY Επίθετο
HAVING COUNT(Παραγγελίες.ΚωδικόςΠαραγγελιών) > 10;
```

Σε αυτό το παράδειγμα, θα απαριθμήσουμε τους εργαζόμενους επίθετο «Νταβόλιο» ή «Φούλερ» εάν έχουν καταχωρίσει παραγγελίες περισσότερες από 25 φορές:

```
SELECT Εργαζόμενοι.Επίθετο, COUNT(Παραγγελίες.ΚωδικόςΠαραγγελίας) AS ΑριθμόςΠαραγγελιών
FROM (Παραγγελίες
INNER JOIN Εργαζόμενοι ON Παραγγελίες.ΚωδικόςΕργαζομένου = Εργαζόμενοι.ΚωδικόςΕργαζομένων)
WHERE Επίθετο = 'Νταβόλιο' OR Επίθετο = 'Φούλερ'
GROUP BY Επίθετο
HAVING COUNT(Παραγγελίες.ΚωδικόςΠαραγγελιών) > 25;
```

Εντολή Select Into στην SQL

Η εντολή SELECT INTO αντιγράφει τα δεδομένα από έναν πίνακα σε έναν νέο πίνακα.

Σύνταξη για αντιγραφή όλων των στηλών σε νέο πίνακα:

```
SELECT *
INTO νέοςπίνακας [IN externaldb]
FROM παλιόςπίνακας
WHERE συνθήκη;
```

Σύνταξη για αντιγραφή μόνο μερικών στηλών σε νέο πίνακα:

```
SELECT στήλη1, στήλη2, ...
INTO νέοςπίνακας [IN externaldb]
FROM παλιόςπίνακας
WHERE συνθήκη;
```

Ο νέος πίνακας θα διατηρήσει τα ονόματα και τους τύπους των στηλών όπως και ο παλιός πίνακας. Μπορείτε να δημιουργήσετε νέες στήλες με τον όρο AS.

Παράδειγμα δημιουργίας εφεδρικού αντιγράφου της στήλης Πελατών:

```
SELECT * INTO ΑντίγραφοΠελατών2017
FROM Πελάτες;
```

Παράδειγμα χρήσης του όρου IN για την αντιγραφή του πίνακα σε νέο πίνακα σε άλλη βάση δεδομένων:

```
SELECT * INTO ΑντίγραφοΠελατών2017 IN 'Backup.mdb'
FROM Πελάτες;
```

Παράδειγμα αντιγραφής μόνο μερικών στηλών σε νέο πίνακα:

```
SELECT ΌνομαΠελάτη, ΌνομαΕπικοινωνίας INTO ΑντίγραφοΠελατών2017
FROM Πελάτες;
```

Παράδειγμα αντιγραφής μόνο των Γερμανών πελατών σε νέο πίνακα:

```
SELECT * INTO ΠελάτεςΓερμανία
FROM Πελάτες
WHERE Χώρα = ‘Γερμανία';
```

Παράδειγμα αντιγραφής δεδομένων από πολλαπλούς πίνακες σε νέο πίνακα:

```
SELECT Πελάτες.ΌνομαΠελάτη, Παραγγελία.ΚωδικόςΠαραγγελίας
INTO ΑντίγραφοΠαραγγελιώνΠελατών2017
FROM Πελάτες
LEFT JOIN Παραγγελίες ON Πελάτες.ΚωδικόςΠελάτη= Παραγγελίες.ΚωδικόςΠελάτη;
```

Η εντολή SELECT INTO μπορεί επίσης να χρησιμοποιηθεί για τη δημιουργία ενός νέου, κενού πίνακα χρησιμοποιώντας το σχήμα ενός άλλου.

Για να το κάνετε αυτό, προσθέστε έναν όρο WHERE που δεν εμφανίζει δεδομένα:

```
>> SELECT * INTO νέοςπίνακας
FROM παλιόςπίνακας
WHERE 1 = 0;
```

Η εντολή Insert Into Select στην SQL

Η εντολή INSERT INTO SELECT αντιγράφει δεδομένα από έναν πίνακα και τα εισάγει σε έναν άλλον. Απαιτεί την αντιστοίχιση των τύπων δεδομένων στον πίνακα πηγής και στόχου.

Σύνταξη για αντιγραφή όλων των στηλών από έναν πίνακα σε έναν άλλο:

```
INSERT INTO πίνακας2
SELECT * FROM πίνακας1
WHERE συνθήκη;
```

Σύνταξη για αντιγραφή μόνο μερικών στηλών από έναν πίνακα σε έναν άλλο:

```
INSERT INTO πίνακας2 (στήλη1, στήλη2, στήλη3, ...)
SELECT στήλη1, στήλη2, στήλη3, ...
FROM πίνακας1
WHERE συνθήκη;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Πίνακας Πελατών - Παράδειγμα εντολής INSERT INTO SELECT

SupplierID	SupplierName	ContactName	Address	City	Postal Code	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

(Πηγή: https://www.w3schools.com/sql/sql_insert_into_select.asp)

Πίνακας Προμηθευτών - Παράδειγμα εντολής INSERT INTO SELECT

(Πηγή: https://www.w3schools.com/sql/sql_insert_into_select.asp)

Θα χρησιμοποιήσουμε τους πίνακες Πελατών και Προμηθευτών, που παρουσιάζονται παραπάνω, στα ακόλουθα παραδείγματα.

Το πρώτο παράδειγμα αντιγράφει Προμηθευτές στον πίνακα Πελατών (σημειώστε ότι οι στήλες που δεν έχουν δεδομένα θα περιέχουν μηδενικές τιμές):

```
INSERT INTO Πελάτες (ΌνομαΠελάτη, Πόλη, Χώρα)
SELECT ΌνομαΠρομηθευτή, Πόλη, Χώρα FROM Προμηθευτές;
```

Αυτό το παράδειγμα αντιγράφει Προμηθευτές στον πίνακα Πελατών για να συμπληρώσει όλες τις στήλες:

```
INSERT INTO Πελάτες (ΌνομαΠελάτη, ΌνομαΕπικοινωνίας, Διεύθυνση, Πόλη, ΤαχυδρομικόςΚώδικας, Χώρα)
SELECT ΌνομαΠρομηθευτή, ΌνομαΕπικοινωνίας, Διεύθυνση, Πόλη, ΤαχυδρομικόςΚώδικας, Χώρα FROM Προμηθευτές;
```

Το τρίτο παράδειγμα αντιγράφει μόνο τους Γερμανούς προμηθευτές στον πίνακα Πελατών:

```
INSERT INTO Πελάτες (ΌνομαΠελάτη, Πόλη, Χώρα)
SELECT ΌνομαΠρομηθευτή, Πόλη, Χώρα FROM Προμηθευτές
WHERE NOT Χώρα='Γερμανία';
```

Εντολή Case στην SQL

Η εντολή CASE περνά από μια σειρά προϋποθέσεων και επιστρέφει μια τιμή όταν πληρούται η πρώτη προϋπόθεση. Σκεφτείτε το σαν μια εντολή με τα if, then, else.

Όταν ένας όρος ισχύει, θα σταματήσει να περνάει μέσα από τον βρόχο. Εάν δεν ισχύει, θα εμφανίσει την τιμή στον όρο ELSE.

Σημειώστε ότι αν δεν υπάρχει ο όρος ELSE και δεν υπάρχουν όροι που να ισχύουν, θα εμφανίσει το NULL.

Σύνταξη:

CASE
WHEN συνθήκη1 THEN αποτέλεσμα1
WHEN συνθήκη2 THEN αποτέλεσμα2
WHEN συθήκηN THEN αποτέλεσμαN
ELSE αποτέλεσμα
END;

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

Πίνακας Παραγγελιών - παράδειγμα του όρου CASE

(Πηγή: https://www.w3schools.com/sql/sql_case.asp)

Στα παρακάτω παραδείγματα, θα χρησιμοποιήσουμε τον πίνακα Παραγγελιών.

Το πρώτο παράδειγμα θα περάσει από μια σειρά συνθηκών και θα εμφανίσει μια τιμή όταν η πρώτη συνθήκη πληρείται:

```
SELECT ΚωδικόςΠαραγγελίας, Ποσότητα,
CASE
WHEN Ποσότητα > 30 THEN 'Ο ποσότητα είναι μεγαλύτερη από 30'
WHEN Ποσότητα = 30 THEN 'Η ποσότητα είναι ίση με 30'
ELSE 'Η ποσότητα είναι μικρότερη από 30'
END AS ΑριθμόςΠοσότητας
FROM ΛεπτομέριεςΠαραγγελίας;
```

Σε αυτό το δεύτερο παράδειγμα, θα ταξινομήσουμε τους πελάτες ανά πόλη. Σημειώστε ότι αν η Πόλη έχει την τιμή NULL, θα ταξινομηθούν ανά Χώρα.

```
SELECT ΌνομαΠελάτη, Πόλη, Χώρα
FROM Πελάτες
ORDER BY
    (CASE
        WHEN Πόλη IS NULL THEN Χώρα
        ELSE Πόλη
END);
```

Λειτουργίες κενών τιμών στην SQL (Null Functions)

Οι λειτουργίες NULL περιλαμβάνουν τα ακόλουθα: IFNULL(), ISNULL(), COALESCE(), and NVL().

Εδώ, θα χρησιμοποιήσουμε τον πίνακα «Προϊόντα»:

P_Id	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

Πίνακας προϊόντων - παράδειγμα λειτουργιών NULL
(Πηγή: https://www.w3schools.com/sql/sql_isnull.asp)

Ας υποθέσουμε ότι η στήλη «ΤεμάχιαΠαραγγελίας» είναι προαιρετική και μπορεί να περιέχει κενές τιμές (NULL).

Παράδειγμα:

```
SELECT ΌνομαΠροϊόντος, ΤιμήΤεμαχίου * (ΤεμάχιαΣεΑπόθεμα +
ΤεμάχιαΠαραγγελίας)

FROM Προϊόντα;
```

Εδώ μπορούμε να δούμε ότι εάν οποιαδήποτε από τις τιμές ΤεμάχιαΠαραγγελίας είναι κενή, το αποτέλεσμα θα είναι επίσης κενό (NULL).

Ας δούμε πώς μπορούμε να λύσουμε αυτό το ζήτημα.

Στο σύστημα MySQL, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση ISNULL() που μας επιτρέπει να επιλέξουμε μια εναλλακτική τιμή αν μια συνάρτηση είναι μηδενική:

```
SELECT ΌνομαΠροϊόντος, ΤιμήΤεμαχίου * (ΤεμάχιαΣεΑπόθεμα +
IFNULL(ΤεμάχιαΠαραγγελίας, 0))
FROM Προϊόντα;
```

Ή μπορούμε να χρησιμοποιήσουμε τη λειτουργία COALESCE() :

```
SELECT ΌνομαΠροϊόντος, ΤιμήΤεμαχίου * (ΤεμάχιαΣεΑπόθεμα +
COALESCE(ΤεμάχιαΠαραγγελίας, 0))
FROM Προϊόντα;
```

Στον διακομιστή SQL, η λειτουργία ISNULL() κάνει το ίδιο πράγμα όπως στο MySQL:

```
SELECT ΌνομαΠροϊόντος, ΤιμήΤεμαχίου * (ΤεμάχιαΣεΑπόθεμα +
IFNULL(ΤεμάχιαΠαραγγελίας, 0))
FROM Προϊόντα;
```

Στη λειτουργία IsNull() στο MS Access η συνάρτηση TRUE(-1) προκύπτει αν η έκφραση είναι μια κενή τιμή, αλλιώς προκύπτει η συνάρτηση FALSE (0):

```
SELECT ΌνομαΠροϊόντος, ΤιμήΤεμαχίου * (ΤεμάχιαΣεΑπόθεμα +
IIF(IsNull(ΤεμάχιαΠαραγγελίας), 0, ΤεμάχιαΠαραγγελίας))
FROM Προϊόντα;
```

Στο σύστημα Oracle, η λειτουργία NVL() κάνει το ίδιο πράγμα:

```
SELECT ΌνομαΠροϊόντος, ΤιμήΤεμαχίου * (ΤεμάχιαΣεΑπόθεμα +
NVL(ΤεμάχιαΠαραγγελίας, 0))
FROM Προϊόντα;
```

Σχόλια στην SQL (Comments)

Τα σχόλια SQL εξηγούν τις ενότητες εντολών στην SQL ή εμποδίζουν την εκτέλεσή τους.

Σημειώστε ότι τα παραδείγματα σε αυτήν την ενότητα δεν υποστηρίζονται στο Firefox και το Microsoft Edge, τα οποία είναι βάσεις δεδομένων της Microsoft Access. Τα σχόλια (comments) γενικά δεν υποστηρίζονται στις βάσεις δεδομένων της Microsoft Access.

Τα σχόλια μιας γραμμής στην SQL ξεκινούν με - - (δύο παύλες):

```
--Select all:
SELECT * FROM Πελάτες;
```

Ή μπορεί να χρησιμοποιηθεί με αυτόν τον τρόπο για να αγνοήσει το τέλος της γραμμής:

```
SELECT * FROM Πελάτες -- WHERE Πόλη='Βερολίνο';
```

Ή για να αγνοήσετε μια πρόταση:

```
--SELECT * FROM Πελάτες;
SELECT * FROM Προϊόντα;
```

Τα σχόλια πολλαπλών γραμμών ξεκινούν με /* και τελειώνουν με */. Οποιοδήποτε κείμενο γραφτεί μεταξύ αυτών των δύο θα αγνοηθεί.

Παράδειγμα:

```
/*Επιλογή όλων των στηλών
```

όλων των εγγραφών
στον Πίνακα Πελατών:*/
SELECT * FROM Πελάτες;

Για να αγνοήσετε μέρος μιας πρότασης, μπορείτε επίσης να χρησιμοποιήσετε το/**/.

Παράδειγμα 1:
SELECT ΌνομαΠελάτη, /*Πόλη,*/ Χώρα FROM Πελάτες;

Παράδειγμα 2:
SELECT * FROM Πελάτες WHERE (ΌνομαΠελάτη LIKE 'Λ%'
OR ΌνομαΠελάτη LIKE 'Ρ%' /*OR ΌνομαΠελάτη LIKE 'Σ%'
OR ΌνομαΠελάτη LIKE 'Τ%'*/ OR ΌνομαΠελάτη LIKE 'Ω%')
AND Χώρα='ΗΠΑ'
ORDER BY ΌνομαΠελάτη;

Τελεστές στην SQL

Αριθμητικοί τελεστές που χρησιμοποιούνται στην SQL:

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

(Πηγή: https://www.w3schools.com/sql/sql_operators.asp)

Οι τελεστές Bitwise που χρησιμοποιούνται στην SQL:

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

(Πηγή: https://www.w3schools.com/sql/sql_operators.asp)

Οι τελεστές σύγκρισης που χρησιμοποιούνται στην SQL:

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

(Πηγή: https://www.w3schools.com/sql/sql_operators.asp)

Σύνθετοι τελεστές που χρησιμοποιούνται στην SQL:

Operator	Description
+=	Add equals
-=	Subtract equals
*=	Multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals
|*=	Bitwise OR equals

(Πηγή: https://www.w3schools.com/sql/sql_operators.asp)

Λογικοί τελεστές που χρησιμοποιούνται στην SQL:

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

(Πηγή: https://www.w3schools.com/sql/sql_operators.asp)

4.2. Βάση δεδομένων στην SQL

Όπως έχουμε αναφέρει στην προηγούμενη υποενότητα που ήταν αφιερωμένη στις βασικές λειτουργίες που χρησιμοποιούνται στην SQL, αυτή η γλώσσα προγραμματισμού χρησιμοποιείται κυρίως για σχεσιακές βάσεις δεδομένων. Ως εκ τούτου, σε αυτή την υποενότητα, θα μάθουμε πώς να δημιουργήσουμε μια βάση δεδομένων, να την τροποποιήσουμε και να την χειριστούμε μέσω της SQL.

Ας ξεκινήσουμε από τις απλές λειτουργίες, προχωρώντας σε ελαφρώς πιο περίπλοκες.

Δημιουργία Βάσης Δεδομένων στην SQL

Η εντολή CREATE DATABASE **δημιουργεί μια νέα βάση δεδομένων στην SQL.**

Σύνταξη:

CREATE DATABASE ΌνομαΒάσηςΔεδομένων;

Σημείωση: Να θυμάστε πάντα ότι το όνομα της βάσης δεδομένων πρέπει να είναι μοναδικό εντός του σχεσιακού Συστήματος Διαχείρισης Βάσεων Δεδομένων (ΣΒΔΒ) που χρησιμοποιείτε και βεβαιωθείτε ότι έχετε δικαιώματα διαχειριστή πριν δημιουργήσετε οποιαδήποτε βάση δεδομένων.

Ας πούμε ότι θέλετε να δημιουργήσετε μια δοκιμαστική βάση δεδομένων. Θα χρησιμοποιούσατε την ακόλουθη εντολή:

CREATE DATABASE δοκιμαστικήΒΔ;

Διαγραφή Βάσης Δεδομένων στην SQL

Η εντολή DROP DATABASE διαγράφει μια υπάρχουσα βάση δεδομένων στην SQL.

DROP DATABASE ΌνομαΒάσηςΔεδομένων;

Πριν διαγράψετε τη βάση δεδομένων, βεβαιωθείτε ότι δεν χρειάζεστε καμία από τις πληροφορίες που περιέχει, καθώς τη διαγράφει εντελώς.

Θυμάστε τη βάση δεδομένων που μόλις δημιουργήσαμε με την ονομασία «δοκιμαστικήΒΔ»; Τώρα θα την διαγράψουμε.

Παράδειγμα:

DROP DATABASE δοκιμαστικήΒΔ;

Δημιουργία αντιγράφου της Βάσης Δεδομένων στην SQL

Η εντολή BACKUP DATABASE δημιουργεί ένα πλήρες εφεδρικό αντίγραφο σε μια υπάρχουσα βάση δεδομένων SQL.

Για να χρησιμοποιήσετε αυτή την εντολή, θα πρέπει να γράψετε δύο πράγματα: το όνομα της βάσης δεδομένων και τη διαδρομή του αρχείου.

BACKUP DATABASE ΌνομαΒάσηςΔεδομένων

TO DISK = ‘διαδρομήαρχείου’

Παράδειγμα:

BACKUP DATABASE δοκιμαστικήΒΔ

TO DISK = 'D:\αντίγραφα\δοκιμαστικήΒΔ.bak';

Σημείωση: Για να αποφύγετε τεχνικά προβλήματα, είναι καλύτερο να δημιουργήσετε εφεδρικά αντίγραφα της βάσης δεδομένων σε μια διαφορετική μονάδα δίσκου από εκείνη στην οποία βρίσκεται η υπάρχουσα βάση δεδομένων.

Υπάρχει επίσης μια άλλη επιλογή όπου μπορείτε να εκτελέσετε ένα διαφορικό εφεδρικό αντίγραφο με βάση τις αλλαγές που έχουν γίνει από το τελευταίο πλήρες εφεδρικό αντίγραφο της βάσης δεδομένων. Αυτός ο τύπος εφεδρικών αντιγράφων μειώνει επίσης τον χρόνο δημιουργίας εφεδρικών αντιγράφων.

Για να το κάνετε αυτό, ακολουθήστε αυτήν τη σύνταξη:

BACKUP DATABASE ΌνομαΒάσηςΔεδομένων
TO DISK = 'διαδρομήαρχείου'
WITH DIFFERENTIAL;

Παράδειγμα:

```
BACKUP DATABASE δοκιμαστικήΔΒ
TO DISK = 'D:\αντίγραφα\δοκιμαστικήΔΒ.bak'
WITH DIFFERENTIAL;
```

Δημιουργία πίνακα στην SQL

Η εντολή CREATE TABLE δημιουργεί έναν νέο πίνακα σε μια βάση δεδομένων.

Σύνταξη:

```
CREATE TABLE όνομα_πίνακα (
                στήλη1 τύποςδεδομένων,
                στήλη2 τύποςδεδομένων,
                στήλη3 τύποςδεδομένων,
                …….
);
```

Σε αυτή τη λειτουργία, θα πρέπει να καθορίσετε τα **ονόματα των στηλών** και τον **τύπο των δεδομένων** που θα περιέχει η στήλη.

Υπάρχουν πολλοί τύποι δεδομένων όπως integer, date ή varchar. Ανάλογα με τον τύπο των δεδομένων που θέλετε να αποθηκεύσετε, επιλέγετε την πιο κατάλληλη επιλογή. Για παράδειγμα, αν έχετε μια στήλη με το όνομα «Ημερομηνία Γέννησης», τότε πιθανότατα θα επιλέγατε την Ημερομηνία ως τον τύπο δεδομένων.

Παράδειγμα:

```
CREATE TABLE Άτομα (
                KωδΑτόμου int,
                Επίθετο varchar(255),
                Όνομα varchar(255),
                Διεύθυνση varchar(255),
                Πόλη varchar(255)
);
```

Αυτό το παράδειγμα θα δημιουργήσει έναν πίνακα με το όνομα Άτομα και θα περιέχει 5 στήλες.

Η στήλη ΚωδΑτόμου θα περιέχει έναν ακέραιο αριθμό (int). Οι στήλες Επίθετο, Όνομα, Διεύθυνση και Πόλη θα περιέχουν μέχρι και 255 χαρακτήρες.

Ο πίνακας χωρίς δεδομένα θα μοιάζει κάπως έτσι:

PersonID	LastName	FirstName	Address	City

Κενός Πίνακας - Παράδειγμα CREATE TABLE

(Πηγή: https://www.w3schools.com/sql/sql_create_table.asp)

Μπορείτε επίσης να δημιουργήσετε έναν πίνακα χρησιμοποιώντας έναν άλλο πίνακα και επιλέγοντας ποιες στήλες θέλετε στον νέο πίνακα. Λάβετε υπόψη ότι τα δεδομένα του υπάρχοντος πίνακα θα συμπληρώσουν τις εγγραφές του νέου πίνακα.

Η σύνταξη έχει ως εξής:

CREATE TABLE όνομα_νέου_πίνακα AS

SELECT στήλη1, στήλη2,...

FROM όνομα_υπάρχοντος_πίνακα

WHERE ....;

Όπως μάθμε στην προηγούμενη ενότητα:

- Η εντολή SELECT επιλέγει τις στήλες από τον υπάρχοντα πίνακα,
- Η εντολή FROM καθορίζει το όνομα του υπάρχοντος πίνακα, και
- Ο όρος WHERE μπορεί να χρησιμοποιηθεί αν θέλετε να επιλέξετε ένα σύνολο εγγραφών που πληρούν μια καθορισμένη συνθήκη.

Παράδειγμα:

```
CREATE TABLE ΔοκιμαστικόςΠίνακας AS
SELECT ΌνομαΠελάτη, ΌνομαΕπικοινωνίας
FROM Πελάτες;
```

Διαγραφή πίνακα στην SQL

Είναι παρόμοια με την εντολή DROP DATABASE που είδαμε νωρίτερα, όμως η εντολή DROP TABLE διαγράφει έναν υπάρχοντα πίνακα σε μια βάση δεδομένων.

Να θυμάστε ότι θα πρέπει να βεβαιωθείτε ότι δεν χρειάζεστε καμία από τις πληροφορίες που περιέχονται σε έναν πίνακα πριν τον διαγράψετε.

Σύνταξη:

```
DROP TABLE ΌνομαΠίνακα;
```

Παράδειγμα:

```
DROP TABLE Άτομα;
```

Μπορείτε επίσης να επιλέξετε να διαγράψετε τα δεδομένα που περιέχονται σε έναν πίνακα, αλλά όχι και τον ίδιο τον πίνακα.

Μπορεί να δημιουργήσατε έναν νέο πίνακα από έναν υπάρχοντα πίνακα που έχει τη δομή που θέλετε, αλλά να θέλετε να προσθέσετε νέες εγγραφές εξ ολοκλήρου. Σε αυτή την περίπτωση, μπορείτε να χρησιμοποιήσετε την εντολή TRUNCATE TABLE.

Σύνταξη:

```
TRUNCATE TABLE ΌνομαΠίνακα;
```

Παράδειγμα:

```
TRUNCATE TABLE Άτομα;
```

Τροποποίηση πίνακα στην SQL (ALTER TABLE)

Η λειτουργία ALTER TABLE μπορεί να προσθέσει, να διαγράψει ή να τροποποιήσει στήλες σε έναν υπάρχοντα πίνακα. Επίσης, μπορεί να χρησιμοποιηθεί για την προσθήκη και την διαγραφή περιορισμών σε έναν υπάρχοντα πίνακα.

Ας δούμε πρώτα τη σύνταξη της προσθήκης μιας στήλης:

ALTER TABLE ΌνομαΠίνακα

ADD όνομα_στήλης τύπος δεδομένων;

Αυτή η λειτουργία μοιάζει με τη λειτουργία που χρησιμοποιήσαμε για να δημιουργήσουμε έναν πίνακα, καθώς πρέπει να καθορίσουμε το όνομα της στήλης και τον τύπο των δεδομένων που θα περιέχονται σε αυτή την στήλη.

Παράδειγμα:
ALTER TABLE Πελάτες
ADD Email varchar(255);

Για να διαγράψουμε μια στήλη σε έναν πίνακα, όπως έχουμε δει στο παρελθόν, χρησιμοποιούμε την εντολή DROP.

Σημειώστε ότι ορισμένα συστήματα βάσεων δεδομένων δεν επιτρέπουν στους χρήστες να διαγράψουν μια στήλη.

Σύνταξη:

ALTER TABLE ΌνομαΠίνακα
DROP COLUMN ΌνομαΣτήλης;

Για παράδειγμα, ας διαγράψουμε τη στήλη που δημιουργήσαμε:
ALTER TABLE Πελάτες
DROP COLUMN Email;

Για να αλλάξετε τον τύπο δεδομένων μιας στήλης, μπορείτε να χρησιμοποιήσετε τις ακόλουθες λειτουργίες ανάλογα με το Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων που χρησιμοποιείτε:

- ALTER COLUMN (για SQL Server/MS Access);
- MODIFY COLUMN (για My SQL/ Oracle πριν την έκδοση 10G);
- MODIFY (για την έκδοση 10G Oracle και μεταγενέστερες εκδόσεις).

Σύνταξη:

ALTER TABLE ΌνομαΠίνακα

ALTER COLUMN ΌνομαΣτήλης τύποςδεδομένων;

Σημειώστε ότι η δεύτερη εντολή είναι αυτή που αλλάζει σύμφωνα με το ΣΔΒΔ που χρησιμοποιείτε, μεταξύ ALTER COLUMN, MODIFY COLUMN και MODIFY. Τα υπόλοιπα παραμένουν τα ίδια.

Ας δούμε ένα παράδειγμα για να κατανοήσουμε καλύτερα αυτή την εντολή. Ο παρακάτω πίνακας είναι ο πίνακας «Άτομα» και περιέχει πληροφορίες για διαφορετικούς ανθρώπους.

ID	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Παράδειγμα τροποποίησης πίνακα (ALTER TABLE)
(Πηγή: https://www.w3schools.com/sql/sql_alter.asp)

Για παράδειγμα, ας πούμε ότι θέλαμε να προσθέσουμε μια στήλη με το όνομα «ΗμερομηνίαΓέννησης» σε αυτόν τον πίνακα. Θα χρησιμοποιήσουμε την ακόλουθη εντολή:

ALTER TABLE Άτομα
ADD ΗμερομηνίαΓέννησης ημερομηνία;

Η νέα στήλη που προσθέσαμε στον πίνακα έχει τον τύπο δεδομένων της ημερομηνίας, που σημαίνει ότι αποθηκεύει δεδομένα σε μορφή ημερομηνίας. Παρακάτω, μπορείτε να δείτε τον πίνακα με την προσθήκη της νέας στήλης.

ID	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	
2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

Παράδειγμα τροποποίησης πίνακα (ALTER TABLE)
(Πηγή: https://www.w3schools.com/sql/sql_alter.asp)

Ωστόσο, αν αλλάξατε γνώμη και θέλετε να αλλάξετε τον τύπο δεδομένων της νέας στήλης, τότε μπορείτε να χρησιμοποιήσετε τη λειτουργία ALTER COLUMN.

Για παράδειγμα, μπορούμε να αλλάξουμε τον τύπο της στήλης που προστέθηκε πρόσφατα από ημερομηνία σε έτος. Για να το κάνετε αυτό, χρησιμοποιήστε την ακόλουθη πρόταση:

```
ALTER TABLE Άτομα
ALTER COLUMN ΗμερομηνίαΓέννησης Έτος;
```

Ο τύπος δεδομένων έτους περιέχει ένα έτος σε διψήφιο ή τετραψήφιο μορφότυπο.

Για να διαγράψουμε τη στήλη που μόλις τροποποιήσαμε, χρησιμοποιούμε την εντολή DROP COLUMN.

```
ALTER TABLE Άτομα
DROP COLUMN ΗμερομηνίαΓέννησης;
```

Ο πίνακας μας θα πάρει τη μορφή που είχε στην αρχή.

ID	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Παράδειγμα τροποποίησης πίνακα (ALTER TABLE)
(Πηγή: https://www.w3schools.com/sql/sql_alter.asp)

Περιορισμοί στην SQL (Constraints)

Οι περιορισμοί SQL χρησιμοποιούνται όταν ο πίνακας δημιουργείται με την εντολή CREATE TABLE ή την εντολή ALTER TABLE κατά την τροποποίησή του.

Σύνταξη:

```
CREATE TABLE όνομα_πίνακα(
        στήλη1 τύποςδεδομένων περιορισμός,
        στήλη2 τύποςδεδομένων περιορισμός,
        στήλη3 τύποςδεδομένων περιορισμός,
        ....
);
```

Οι περιορισμοί χρησιμοποιούνται για τον καθορισμό ενός συνόλου κανόνων και περιορισμών που ισχύουν για μια στήλη ή έναν πίνακα. Χρησιμοποιούνται για τη διασφάλιση της ακεραιότητας, της ακρίβειας και της αξιοπιστίας των δεδομένων. Εάν εφαρμόζονται περιορισμοί σε έναν πίνακα, τότε όλες οι στήλες πρέπει να διαμορφώνονται με βάση αυτούς τους περιορισμούς.

Οι ακόλουθοι περιορισμοί είναι αυτοί που χρησιμοποιούνται τις περισσότερες φορές:

- NOT NULL
- UNIQUE

- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT
- CREATE INDEX

Θα εξετάσουμε καθέναν από αυτούς τους περιορισμούς για να εξηγήσουμε τη χρήση και τη σύνταξή τους με παραδείγματα.

Περιορισμοί ύπαρξης τιμής στην SQL (NOT NULL)

Στην SQL, οι στήλες μπορούν να έχουν κενές τιμές από προεπιλογή. Ο περιορισμός NOT NULL χρησιμοποιείται για την αποφυγή των κενών τιμών σε στήλες. Αυτό είναι ιδιαίτερα σημαντικό για να εξασφαλιστεί ότι, όταν προστίθεται μια νέα εγγραφή σε έναν πίνακα, συμπληρώνονται όλα τα απαραίτητα πεδία.

Για παράδειγμα, ας πούμε ότι θέλουμε να δημιουργήσουμε έναν πίνακα με το όνομα «Άτομα» και θέλουμε να διασφαλίσουμε ότι οι στήλες «ΚωδΑτόμου», «Επίθετο» και «Όνομα» δεν θα περιέχουν κενές τιμές:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255) NOT NULL,
        Ηλικία int
);
```

Εάν για κάποιο λόγο, θέλετε να τροποποιήσετε έναν ήδη υπάρχοντα πίνακα για να προσθέσετε περιορισμούς, μπορείτε να χρησιμοποιήσετε την ακόλουθη εντολή:

```
ALTER TABLE Άτομα
MODIFY Ηλικία int NOT NULL;
```

Μοναδικές τιμές στην SQL (Unique)

Ο περιορισμός UNIQUE χρησιμοποιείται για να διασφαλιστεί ότι όλες οι τιμές που αποθηκεύονται σε μια στήλη είναι μοναδικές μεταξύ των σειρών ενός πίνακα. Για να γίνει αυτό σαφέστερο, σκεφτείτε τη μεταβλητή «ΚωδΑτόμου». Για να μην έχουν δύο άτομα την ίδια ταυτότητα, χρησιμοποιούμε τον περιορισμό UNIQUE.

SQL Server / Oracle / MS Access:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL UNIQUE,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int
);
```

My SQL:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int,
UNQUE (Ταυτότητα)
);
```

Όπως βλέπετε υπάρχουν προσαρμογές σχετικά με το που τοποθετείται ο περιορισμός UNIQUE στον κώδικα, ανάλογα με το ΣΣΔΒΔ που χρησιμοποιείτε.

Εάν θέλετε να ονομάσετε ή να θέσετε έναν περιορισμό UNIQUE σε πολλαπλές στήλες, χρησιμοποιήστε τα ακόλουθα:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
```

```
    Όνομα varchar(255),
    Ηλικία int,
CONSTRAINT ΜΠ_Άτομα UNIQUE (Ταυτότητα, Επίθετο)
);
```

Μπορείτε επίσης να προσθέσετε έναν περιορισμό UNIQUE μετά τη δημιουργία του πίνακα χρησιμοποιώντας την εντολή ALTER TABLE που μάθαμε νωρίτερα.

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Άτομα
ADD UNIQUE (ΚωδΑτόμου);
```

Εάν θέλετε επίσης να ονομάσετε και να θέσετε έναν περιορισμό UNIQUE σε πολλές ήδη υπάρχουσες στήλες, χρησιμοποιείτε την ακόλουθη εντολή:

```
ALTER TABLE Άτομα
ADD CONSTRAINT ΜΠ_Άτομα UNIQUE (ΚωδΑτόμου, Επίθετο);
```

Για να διαγράψετε τον περιορισμό UNIQUE, μπορείτε να χρησιμοποιήσετε την ακόλουθη εντολή:

My SQL:

```
ALTER TABLE Άτομα
DROP INDEX ΜΠ_Άτομα;
```

SQL Server/Oracle/ MS Access:

```
ALTER TABLE Άτομα
DROP CONSTRAINT ΜΠ_Άτομα;
```

Πρωτεύον Κλειδί στην SQL (Primary Key)

Ο περιορισμός PRIMARY KEY χρησιμοποιείται για τη μοναδική αναγνώριση κάθε σειράς ή εγγραφής σε έναν πίνακα. Σημειώστε ότι οι περιορισμοί πρέπει να περιέχουν μοναδικές τιμές, ωστόσο δεν μπορούν να περιέχουν κενές τιμές.

Ένας πίνακας μπορεί να έχει μόνο ΕΝΑ πρωτεύον κλειδί το οποίο αυτό μπορεί να αποτελείται από μία ή περισσότερες στήλες.

SQL Server / Oracle / MS Access:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL PRIMARY KEY,
        Επίθετο varchar(255) NOT NULL,
        Όνομα archar(255),
        Ηλικία int
);
```

MySQL:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int,
    PRIMARY KEY (Ταυτότητα)
);
```

Το ακόλουθο παράδειγμα σας επιτρέπει να ονομάσετε και να θέσετε έναν περιορισμό PRIMARY KEY σε πολλαπλές στήλες:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
```

```
    Όνομα varchar(255),
    Ηλικία int,
    CONSTRAINT ΠΚ_Άτομα PRIMARY KEY (Ταυότητα,Επίθετο)
);
```

Σημειώστε ότι υπάρχει ένα μόνο PRIMARY KEY, ωστόσο η τιμή του περιλαμβάνει δύο στήλες.

Μπορείτε επίσης να δημιουργήσετε έναν περιορισμό PRIMARY KEY σε έναν υπάρχοντα πίνακα χρησιμοποιώντας την ακόλουθη εντολή:

```
ALTER TABLE Άτομα
ADD PRIMARY KEY (ΚωδΑτόμου);
```

Για να προσθέσετε και να θέσετε έναν περιορισμό PRIMARY KEY σε έναν υπάρχοντα πίνακα, χρησιμοποιήστε την ακόλουθη εντολή:

```
ALTER TABLE Άτομα
ADD CONSTRAINT ΠΚ_Άτομα PRIMARY KEY (ΚωδΑτόμου, Επίθετο);
```

Για να διαγράψετε έναν περιορισμό PRIMARY KEY, χρησιμοποιήστε τις ακόλουθες εντολές σύμφωνα με το ΣΣΔΒΔ που χρησιμοποιείτε.

MySQL:

```
ALTER TABLE Άτομα
DROP PRIMARY KEY;
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Άτομα
DROP PRIMARY KEY;
```

Ξένο Κλειδί στην SQL (Foreign Key)

Το FOREING KEY αντιπροσωπεύει τις στήλες ενός πίνακα που συνδέονται με έναν περιορισμό PRIMARY KEY σε έναν άλλο πίνακα. Ο πίνακας που έχει τον περιορισμό FOREIGN KEY ονομάζεται child table, ενώ ο πίνακας που έχει το primary key ονομάζεται πίνακας αναφοράς ή parent table.

Αυτός ο τύπος περιορισμού χρησιμοποιείται για να αποτρέψει οποιεσδήποτε ενέργειες που θα κατέστρεφαν τους δεσμούς μεταξύ του parent table και του child table.

Ας εξετάσουμε τους παρακάτω πίνακες:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Πίνακας Ατόμων – Παράδειγμα του περιορισμού FOREING KEY
(Πηγή: https://www.w3schools.com/sql/sql_foreignkey.asp)

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Πίνακας Παραγελιών – Παράδειγμα περιορισμού FOREING KEY
(Πηγή: https://www.w3schools.com/sql/sql_foreignkey.asp)

Αυτοί οι δύο πίνακες συνδέονται με τη στήλη «ΑναγνωριστικόΑτόμου» (PersonID) που βρίσκεται και στους δύο πίνακες. Τώρα, το primary key βρίσκεται στον πίνακα ΄Ατομα και το foreign key στη στήλη «ΑναγνωριστικόΑτόμου» στον πίνακα Παραγγελίες.

Ο περιορισμός FOREIGN KEY λειτουργεί εμποδίζοντας την εισαγωγή μη έγκυρων δεδομένων στη στήλη του FOREIGN KEY επειδή συνδέεται με τον parent table και οι τιμές του πρέπει να είναι πανομοιότυπες.

Για να χρησιμοποιήσετε τον περιορισμό FOREIGN KEY κατά τη δημιουργία ενός πίνακα, μπορείτε να χρησιμοποιήσετε την ακόλουθη εντολή ανάλογα με το ΣΣΔΒΔ που χρησιμοποιείτε.

SQL Server / Oracle / MS Access:

```
CREATE TABLE Παραγγελίες (
        ΚωδικόςΠαραγγελίας int NOT NULL PRIMARY KEY,
        ΑριθμόςΠαραγγελίας int NOT NULL,
        ΚωδΑτόμου int FOREIGN KEY REFERENCES Άτομα(ΚωδΑτόμου)
);
```

My SQL:

```
CREATE TABLE Παραγγελίες (
        ΚωδικόςΠαραγγελίας int NOT NULL,
        ΑριθμόςΠαραγγελίας int NOT NULL,
        ΚωδΑτόμου int,
        PRIMARY KEY (ΚωδικόςΠαραγγελίας),
        FOREIGN KEY (ΚωδΑτόμου) REFERENCES Άτομα(ΚωδΑτόμου)
);
```

Αυτή η εντολή συνέδεσε τον πίνακα Παραγγελιών με τον πίνακα Ατόμων χρησιμοποιώντας τον περιορισμό FOREIGN KEY με βάση τη στήλη ΚωδΑτόμου.

Περιορισμός ελέγχου τιμής στην SQL (CHECK)

Ο περιορισμός CHECK χρησιμοποιείται για να καθορίσει τις τιμές που επιτρέπονται σε μια στήλη ή σε ορισμένες στήλες ενός πίνακα με βάση τις τιμές που βρίσκονται σε άλλες στήλες της ίδιας σειράς.

Παράδειγμα περιορισμού CHECK στη λειτουργία CREATE TABLE

Το ακόλουθο παράδειγμα χρησιμοποιείται για να διασφαλιστεί ότι ένα άτομο δεν είναι κάτω από την ηλικία των 18 ετών, προσθέτοντας τον περιορισμό CHECK στη στήλη «Ηλικία».

MySQL:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int,
    CHECK (Age>=18)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int CHECK (Age>=18)
);
```

Εάν θέλετε να καθορίσετε έναν περιορισμό CHECK και να χρησιμοποιήσετε τον περιορισμό σε πολλαπλές στήλες, μπορείτε να χρησιμοποιήσετε την ακόλουθη εντολή:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int,
       Πόλη varchar(255),
       CONSTRAINT ΕΛΓΧ_Άτομο CHECK (Ηλικία>=18 AND Πόλη= 'Σάντνες')
);
```

Παράδειγμα περιορισμού CHECK στην εντολή ALTER TABLE

Για να καθορίσετε έναν περιορισμό σε έναν ήδη υπάρχοντα πίνακα, χρησιμοποιήστε την ακόλουθη εντολή.

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Άτομα
ADD CHECK (Ηλικία>=18);
```

Για να καθορίσετε έναν περιορισμό και να τον προσθέσετε σε πολλαπλές στήλες, μπορείτε να χρησιμοποιήσετε:

```
ALTER TABLE Άτομα
ADD CONSTRAINT ΕΛΓΧ_Άτομα CHECK (Ηλικία>=18 AND Πόλη= 'Σάντνες');
```

Παράδειγμα εντολής DROP ενός περιορισμού CHECK

Για να διαγράψετε έναν περιορισμό CHECK, μπορείτε να χρησιμοποιήσετε τα ακόλουθα σύμφωνα με το ΣΣΔΒΔ που χρησιμοποιείτε.

SQL Server / Oracle / MS Access:

```
ALTER TABLE Άτομα
```

DROP CONSTRAINT ΕΛΓΧ_ΗλικίαΑτόμου;

MySQL:

ALTER TABLE Άτομα

DROP CHECK ΕΛΓΧ_ΗλικίαΑτόμου;

Περιορισμός προεπιλεγμένων τιμών στην SQL (DEFAULT)

Ο περιορισμός DEFAULT χρησιμοποιείται για να καθορίσει μια προεπιλεγμένη τιμή για μια στήλη. Εάν δεν έχουν καθοριστεί άλλες τιμές, η προεπιλεγμένη τιμή θα προστεθεί σε όλες τις νέες εγγραφές.

Παράδειγμα περιορισμού DEFAULT στη λειτουργία CREATE TABLE

Το ακόλουθο παράδειγμα προσθέτει μια προεπιλεγμένη τιμή στη στήλη Πόλη όταν δημιουργείται ο πίνακας Άτομα:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
        Ηλικία int,
        Πόλη varchar(255) DEFAULT ‘Σάντνες'
);
```

Αυτός ο περιορισμός μπορεί επίσης να χρησιμοποιηθεί για την εισαγωγή τιμών με τελεστές όπως το GETDATE():

```
CREATE TABLE Παραγγελίες (
        ΚωδικόςΠαραγγελίας int NOT NULL,
```

```
        ΑρΠαραγγελίας int NOT NULL,
        ΗμερομηνίαΠαραγγελίας ημερομηνία DEFAULT GETDATE()
);
```

Παράδειγμα περιορισμού DEFAULT με την εντολή ALTER TABLE

Σε αυτό το παράδειγμα, η στήλη «Πόλη» χρησιμοποιείται για τον καθορισμό ενός περιορισμού DEFAULT όταν τροποποιούμε έναν ήδη υπάρχοντα πίνακα.

MySQL:

```
ALTER TABLE Άτομα
ALTER Πόλη SET DEFAULT 'Σάντνες';
```

SQL Server:

```
ALTER TABLE Άτομα
ADD CONSTRAINT προεπ_Πόλη
DEFAULT 'Σάντνες' FOR Πόλη;
```

MS Access:

```
ALTER TABLE Άτομα
ALTER COLUMN ΄Πόλη SET DEFAULT 'Σάντνες';
```

Oracle:

```
ALTER TABLE Άτομα
MODIFY Πόλη DEFAULT 'Σάντνες';
```

Παράδειγμα κατάργησης ενός περιορισμού DEFAULT με την εντολή DROP

MySQL:

```
ALTER TABLE Άτομα
ALTER Πόλη DROP DEFAULT;
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Άτομα
ALTER COLUMN Πόλη DROP DEFAULT;
```

Δείκτες στην SQL (Index)

Η εντολή CREATE INDEX δημιουργεί έναν δείκτη σε έναν πίνακα. Οι δείκτες είναι χρήσιμοι όταν θέλετε να ανακτήσετε δεδομένα πιο γρήγορα.

Σημειώστε ότι οι πίνακες με δείκτες χρειάζονται περισσότερο χρόνο για να ενημερωθούν σε σύγκριση με τους πίνακες χωρίς δείκτες. Ως εκ τούτου, προτείνεται να δημιουργηθούν μόνο δείκτες σε στήλες όπου αναζητούνται δεδομένα συχνά.

Για την εντολή CREATE INDEX σε έναν πίνακα όπου επιτρέπονται διπλές τιμές, χρησιμοποιήστε την ακόλουθη σύνταξη:

```
CREATE INDEX όνομα_δείκτη
ON όνομα_πίνακα (στήλη1, στήλη2, ...);
```

Για την εντολή CREATE UNIQUE INDEX σε έναν πίνακα όπου επιτρέπονται διπλές τιμές, χρησιμοποιήστε την ακόλουθη σύνταξη:

```
CREATE UNIQUE INDEX όνομα_δείκτη
ON όνομα_πίνακα (στήλη1, στήλη2, ...);
```

Λάβετε υπόψη ότι η δημιουργία δεικτών ποικίλλει από βάση δεδομένων σε βάση δεδομένων, οπότε πάντα να ελέγχετε τη σύνταξη όταν δημιουργείτε έναν δείκτη στη βάση δεδομένων σας.

Παραδείγματα της εντολής CREATE INDEX

Σε αυτό το παράδειγμα, δημιουργούμε έναν δείκτη στη στήλη Επίθετο καθορίζοντας το όνομα δκτς_επίθετο:

```
CREATE INDEX δκτς_επίθετο
ON Άτομα (Επίθετο);
```

Για να δημιουργήσετε έναν δείκτη σε ένα συνδυασμό στηλών, χρησιμοποιήστε την ακόλουθη εντολή:

```
CREATE INDEX δκτς_όνομα
ON Άτομα (Επίθετο, Όνομα);
```

Αν θέλετε, μπορείτε να προσθέσετε περισσότερες στήλες στην παρένθεση.

Παραδείγματα της εντολής DROP INDEX για τη διαγραφή ενός δείκτη

Εάν θέλετε να διαγράψετε έναν δείκτη, χρησιμοποιήστε την ακόλουθη εντολή σύμφωνα με το ΣΣΔΒΔ που χρησιμοποιείτε.

MS Access:

```
DROP INDEX όνομα_δείκτη ON όνομα_πίνακα;
```

SQL Server:

```
DROP INDEX όνομα_πίνακα.όνομα_δείκτη;
```

DB2/Oracle:

```
DROP INDEX όνομα_δείκτη;
```

MySQL:

```
ALTER TABLE όνομα_πίνακα
DROP INDEX όνομα_δείκτη;
```

Αυτόματη Αύξηση Τιμής στην SQL (Auto-Increment)

Η αυτόματη αύξηση χρησιμοποιείται για την αυτόματη δημιουργία μοναδικών αριθμών όταν εισάγεται μια νέα εγγραφή σε έναν πίνακα. Αυτό χρησιμοποιείται συνήθως σαν Primary key προκειμένου να διασφαλιστεί ότι κανένα άτομο δεν έχει την ίδια ταυτότητα.

Αυτή η λειτουργία χρησιμοποιεί διαφορετική σύνταξη στα συστήματα MySQL, SQL Server, Access και Oracle. Ως εκ τούτου, θα εξετάσουμε καθένα από αυτά για να εξηγήσουμε πώς να χρησιμοποιήσετε την αυτόματη αύξηση τιμών.

MySQL:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int NOT NULL AUTO_INCREMENT,
         Επίθετο varchar(255) NOT NULL,
         Όνομα varchar(255),
         Ηλικία int,
        PRIMARY KEY (ΚωδΑτόμου)
);
```

Στο σύστημα MySQL, η εντολή AUTO_INCREMENT προσθέτει τη λειτουργία αυτόματης αύξησης και η τιμή ορίζεται 1 από προεπιλογή και αυξάνεται κατά 1 κάθε φορά.

Εάν θέλετε η ακολουθία να ξεκινήσει από διαφορετική τιμή, χρησιμοποιήστε την ακόλουθη εντολή:

```
ALTER TABLE Άτομα AUTO_INCREMENT=100;
```

Εάν εισαγάγετε μια νέα εγγραφή στον πίνακα «Άτομα», δεν θα χρειαστεί να καθορίσετε μια τιμή για τη στήλη «ΚωδΑτόμου», καθώς θα δημιουργηθεί αυτόματα:

```
INSERT INTO Άτομα (Όνομα, Επίθετο)
VALUES (‘Λαρς','Μόνσεν');
```

SQL Server

Ακολουθούμε το ίδιο παράδειγμα όπως είδαμε στις προηγούμενες διαφάνειες για το MySQL, όπου χρησιμοποιούμε τη στήλη «ΚωδΑτόμου» ως το primary key στον πίνακα «Άτομα»:

```
CREATE TABLE Άτομα (
        ΚωδΑτόμου int IDENTITY(1,1) PRIMARY KEY,
         Επίθετο varchar(255) NOT NULL,
         Όνομα varchar(255),
         Ηλικία int
);
```

Στο σύστημα SQL Server, η εντολή της αυτόματης αύξησης τιμών χρησιμοποιεί τη λέξη-κλειδί IDENTIFY για να ενεργοποιηθεί. Οι δύο τιμές στην παρένθεση υποδεικνύουν (την αρχική τιμή, προσθέτοντας μια νέα τιμή για κάθε νέα εγγραφή). Ξεκινά από το 1 και αυξάνεται κατά 1 κάθε φορά που εισάγεται μια νέα εγγραφή.

Σε περίπτωση που θέλετε να αλλάξετε την αρχική τιμή σε 10 και να προσθέτετε 5 κάθε φορά που προστίθεται μια νέα εγγραφή, θα πρέπει να την γράψετε ως εξής: IDENTIFY (10,5).

Κατά την εισαγωγή νέων εγγραφών, δεν χρειάζεται να καθορίσετε την τιμή στη στήλη ΚωδΑτόμου. Θα εμφανιστεί αυτόματα όπως στο παραπάνω παράδειγμα.

MS Access

```
CREATE TABLE Άτομα (
ΚωδΑτόμου AUTOINCREMENT PRIMARY KEY,
        Επίθετο varchar(255) NOT NULL,
        Όνομα varchar(255),
```

Ηλικία int

);

Το MS Access χρησιμοποιεί τη λέξη-κλειδί AUTOINCREMENT για να ενεργοποιήσει τη λειτουργία αυτόματης αύξησης τιμών. Η αρχική τιμή είναι το 1 και αυξάνεται κατά 1 κάθε φορά που προστίθεται μια νέα εγγραφή, όπως και στις άλλες δύο περιπτώσεις.

Μπορείτε να καθορίσετε διαφορετικές τιμές όπως 10 για την αρχική τιμή και να αυξάνεται κατά 5 με κάθε προσθήκη με την εντολή AUTOINCREMENT(10,5).

Και πάλι, σημειώστε ότι κάθε φορά που προσθέτουμε μια νέα εγγραφή, δεν χρειάζεται να καθορίσουμε τον Κωδικό Ατόμου. Εμφανίζεται αυτόματα.

Oracle

Στο σύστημα Oracle, ο κώδικας για την αυτόματη αύξηση τιμών είναι λίγο πιο περίπλοκος. Για να δημιουργήσετε ένα πεδίο αυτόματης αύξησης, πρέπει να δημιουργήσετε μια ακολουθία αριθμών:

```
CREATE SEQUENCE ακ_ατόμου
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 10;
```

Αυτή η ακολουθία δημιουργεί ένα αντικείμενο ακολουθίας που ονομάζεται «ακ_ατόμου», ορίζει την ελάχιστη τιμή από την οποία ξεκινά (η οποία είναι 1 σε αυτήν την περίπτωση), έπειτα καθορίζει την αύξηση κατά 1. Η προσωρινή μνήμη καθορίζει πόσες τιμές ακολουθίας θα πρέπει να αποθηκεύονται στη μνήμη για γρηγορότερη πρόσβαση.

Σε αντίθεση με τα προηγούμενα παραδείγματα, για να εισαγάγετε μια νέα εγγραφή στον πίνακα Άτομα, θα πρέπει να χρησιμοποιήσετε τη λειτουργία nextval. Αυτή η λειτουργία χρησιμοποιείται για να ανακτήσει την επόμενη τιμή από το αντικείμενο ακολουθίας που δημιουργήσαμε.

INSERT INTO Άτομα (ΚωδΑτόμου, Όνομα, Επίθετο)

VALUES (ακ_ατόμου.nextval,'Λαρς','Μόνσεν');

Εδώ, μπορούμε να δούμε ότι η στήλη ΚωδΑτόμου επιλέγεται για να εκχωρηθεί ο επόμενος αριθμός από το αντικείμενο ακολουθίας που δημιουργήσαμε που ονομάζεται «ακ_ατόμου».

Ημερομηνίες στην SQL

Μια από τις μεγαλύτερες προκλήσεις όταν εργάζεστε με ημερομηνίες (date) είναι να διασφαλίσετε ότι η μορφή της ημερομηνίας που προσπαθείτε να εισάγετε είναι η ίδια με τη μορφή της στήλης ημερομηνίας στη βάση δεδομένων.

Είναι σημαντικό να σημειωθεί ότι τα δεδομένα που περιέχουν μόνο ημερομηνίες θα λειτουργήσουν όπως αναμένεται στα ερωτήματα. Ωστόσο, αν συμπεριλαμβάνεται και η ώρα (time), τα πράγματα γίνονται λίγο πιο περίπλοκα.

Τύποι χρονολογικών δεδομένων στο MySQL:

- DATE (Ημερομηνία) - μορφότυπο EEEE-ΜΜ-ΗΗ
- DATETIME (Ημερομηνία και Ώρα) - μορφότυπο: EEEE-ΜΜ-ΗΗ ΩΩ: ΛΛ:ΔΔ
- TIMESTAMP (Χρονοσήμανση) - μορφή: EEEE-ΜΜ-ΗΗ ΩΩ: ΛΛ:ΔΔ
- YEAR (Έτος) - Μορφή EEEE ή ΕΕ

Τύποι χρονολογικών δεδομένων στο SQL Server:

- DATE – μορφή: EEEE-ΜΜ-ΗΗ
- DATETIME- μορφή: EEEE-ΜΜ-ΗΗ ΩΩ: ΛΛ:ΔΔ
- SMALLDATETIME - μορφή: EEEE-ΜΜ-ΗΗ ΩΩ: ΛΛ:ΔΔ
- TIMESTAMP - μορφή: ένας μοναδικός αριθμός

Λάβετε υπόψη ότι οι τύποι δεδομένων επιλέγονται όταν δημιουργείτε έναν νέο πίνακα στη βάση δεδομένων σας.

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11
2	Camembert Pierrot	2008-11-09
3	Mozzarella di Giovanni	2008-11-11
4	Mascarpone Fabioli	2008-10-29

Πίνακας παραγγελιών - Παράδειγμα ημερομηνιών
(Πηγή: https://www.w3schools.com/sql/sql_dates.asp)

Θα χρησιμοποιήσουμε τον πίνακα Παραγγελιών στο παράδειγμά μας για να επιλέξουμε τις εγγραφές με ημερομηνία παραγγελίας “2008-11-11”.

Παράδειγμα:

```
SELECT *
FROM Παραγγελίες
WHERE ΗμερομηνίαΠαραγγελίας='2008-11-11’;
```

Το αναμενόμενο αποτέλεσμα θα είναι κάπως έτσι:

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11
3	Mozzarella di Giovanni	2008-11-11

Αποτέλεσμα ερωτήματος Ημερομηνιών Παραγγελίας - Παράδειγμα Ημερομηνιών
(Πηγή: https://www.w3schools.com/sql/sql_dates.asp)

Σημειώστε ότι δύο ημερομηνίες μπορούν να συγκριθούν εύκολα όταν δεν υπάρχει Χρονοσήμανση (TIMESTAMP).

Ας υποθέσουμε ότι έχετε τον πίνακα Παραγγελιών, αλλά με μια χρονική σήμανση στη στήλη Ημερομηνία Παραγγελίεας (OrderDate).

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11 13:23:44
2	Camembert Pierrot	2008-11-09 15:45:21
3	Mozzarella di Giovanni	2008-11-11 11:12:01
4	Mascarpone Fabioli	2008-10-29 14:56:59

Πίνακας παραγγελιών με χρονοσήμανση - Παράδειγμα Ημερομηνιών
(Πηγή: https://www.w3schools.com/sql/sql_dates.asp)

Εδώ, εάν προσπαθήσατε να χρησιμοποιήσετε το ίδιο ερώτημα με αυτό που χρησιμοποιήσαμε παραπάνω θα προκύψει το παρακάτω:

```
SELECT *
FROM Παραγγελίες
WHERE Ημερομηνία Παραγγελίας='2008-11-11';
```

Δεν θα πάρετε κανένα αποτέλεσμα, επειδή το ερώτημα δεν λαμβάνει υπόψη τη χρονοσήμανση. Συνιστάται να μην χρησιμοποιείτε χρονοσημάνσεις, εκτός αν είναι απολύτως απαραίτητο.

Προβολές στην SQL (VIEWS)

Στην SQL, μια προβολή (view) είναι ένας εικονικός πίνακας του συνόλου αποτελεσμάτων που δημιουργείται από ένα συγκεκριμένο ερώτημα. Μια προβολή είναι χρήσιμη όταν θέλετε να προβάλετε και να παρουσιάσετε δεδομένα μέσω ενός συνδυασμού πινάκων.

Σύνταξη:

```
CREATE VIEW όνομα_προβολής AS
SELECT στήλη1, στήλη2, ...
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Σημειώστε ότι μια προβολή εμφανίζει πάντα ενημερωμένα δεδομένα, καθώς η βάση δεδομένων αναδημιουργεί τον εικονικό πίνακα, κάθε φορά που οι χρήστες δημιουργούν το αντίστοιχο ερώτημα.

Παράδειγμα για να δημιουργήσετε ερώτημα με όλους τους πελάτες από τη Βραζιλία:

```
CREATE VIEW [Πελάτες από Βαζιλία] AS
SELECT ΌνομαΠελάτη, ΌνομαΕπικοινωνίας
FROM Πελάτες
WHERE Country = ‘Βραζιλία';
```

Για να προβάλετε το ερώτημα:

```
SELECT * FROM [Πελάτες από Βραζιλία];
```

Παράδειγμα δημιουργίας προβολής που επιλέγει κάθε προϊόν στον πίνακα Προϊόντων με τιμή υψηλότερη από τη μέση τιμή:

```
CREATE VIEW [Προϊόντα με υψηλότερη τιμή από τη μέση τιμή] AS
SELECT ΌνομαΠροϊόντος, Τιμή
FROM Προϊόντα
WHERE Τιμή > (SELECT AVG(Τιμή) FROM Προϊόντα);
```

Για να υποβάλετε ένα ερώτημα σχετικά με την παραπάνω προβολή:

```
SELECT * FROM [Προϊόντα με υψηλότερη τιμή από τη μέση τιμή];
```

Για να ενημερώσετε μια προβολή, χρησιμοποιήστε την εντολή CREATE OR REPLACE VIEW:

```
CREATE OR REPLACE VIEW όνομα_προβολής AS
SELECT στήλη1, στήλη2, ...
FROM όνομα_πίνακα
WHERE συνθήκη;
```

Για να προσθέσετε τη στήλη «Πόλη» στην προβολή με όνομα Πελάτες από Βραζιλία που δημιουργήσαμε νωρίτερα:

```
CREATE OR REPLACE VIEW [Πελάτες από Βραζιλία] AS
SELECT ΌνομαΠελάτη, ΌνομαΕπικοινωνίας, Πόλη
FROM Πελάτες
WHERE Χώρα = 'Βραζιλία';
```

Για να διαγράψετε μια προβολή, χρησιμοποιήστε την εντολή DROP VIEW:

```
DROP VIEW όνομα_προβολής;
```

Για να διαγράψετε την προβολή «Πελάτες από Βραζιλία»:

```
DROP VIEW [Πελάτες από Βραζιλία];
```

Τύποι δεδομένων στην SQL

Γενικά, κάθε στήλη σε έναν πίνακα απαιτεί ένα όνομα και έναν τύπο δεδομένων.

Ένας προγραμματιστής της SQL θα πρέπει να αποφασίσει τον τύπο των δεδομένων που θα αποθηκευτούν μέσα σε κάθε στήλη κατά τη δημιουργία ενός πίνακα. Ο τύπος δεδομένων χρησιμοποιείται στην SQL για την κατανόηση των δεδομένων που θα περιέχονται σε κάθε στήλη και επίσης πώς θα αλληλοεπιδρά με αυτά τα δεδομένα.

Λάβετε υπόψη ότι οι τύποι δεδομένων μπορεί να έχουν διαφορετικά ονόματα σε κάθε βάση δεδομένων. Πρέπει πάντοτε να ελέγχετε την τεκμηρίωση, ακόμη και αν το όνομα είναι το ίδιο, καθώς άλλες λεπτομέρειες μπορεί να είναι διαφορετικές, όπως το μέγεθος.

Τύποι δεδομένων στο σύστημα MySQL (Έκδοση 8.0)

Το σύστημα MySQL έχει τρεις βασικούς τύπους δεδομένων: συμβολοσειρά, αριθμητικός τύπος και ημερομηνίας/ώρας.

Τύποι δεδομένων συμβολοσειράς

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

Αριθμητικοί τύποι δεδομένων

Data type	Description
BIT(size)	A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.
TINYINT(size)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(size)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255)
MEDIUMINT(size)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255)
INT(size)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255)
INTEGER(size)	Equal to INT(size)
BIGINT(size)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255)
FLOAT(size, d)	A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(p)	A floating point number. MySQL uses the p value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(size, d)	A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter
DOUBLE PRECISION(size, d)	
DECIMAL(size, d)	An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0.
DEC(size, d)	Equal to DECIMAL(size,d)

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

Τύποι δεδομένων ημερομηνίας και ώρας

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(fsp)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(fsp)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp**)**

Τύποι δεδομένων στο σύστημα SQL Server:

Τύποι δεδομένων συμβολοσειράς

Data type	Description	Max size	Storage
char(n)	Fixed width character string	8,000 characters	Defined width
varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string	2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string	4,000 characters	
nvarchar(max)	Variable width Unicode string	536,870,912 characters	
ntext	Variable width Unicode string	2GB of text data	
binary(n)	Fixed width binary string	8,000 bytes	
varbinary	Variable width binary string	8,000 bytes	
varbinary(max)	Variable width binary string	2GB	
image	Variable width binary string	2GB	

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

Αριθμητικοί τύποι δεδομένων

Data type	Description	Storage
bit	Integer that can be 0, 1, or NULL	
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 -1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
numeric(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 -1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

Τύποι δεδομένων ημερομηνίας και ώρας

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable	

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

Άλλοι τύποι δεδομένων

Data type	Description
sql_variant	Stores up to 8,000 bytes of data of various data types, except text, ntext, and timestamp
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	Stores XML formatted data. Maximum 2GB
cursor	Stores a reference to a cursor used for database operations
table	Stores a result-set for later processing

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

Τύποι δεδομένων στο σύστημα MS Access

Data type	Description	Storage
Text	Use for text or combinations of text and numbers. 255 characters maximum	
Memo	Memo is used for larger amounts of text. Stores up to 65,536 characters. **Note:** You cannot sort a memo field. However, they are searchable	
Byte	Allows whole numbers from 0 to 255	1 byte
Integer	Allows whole numbers between -32,768 and 32,767	2 bytes
Long	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
Single	Single precision floating-point. Will handle most decimals	4 bytes
Double	Double precision floating-point. Will handle most decimals	8 bytes
Currency	Use for currency. Holds up to 15 digits of whole dollars, plus 4 decimal places. **Tip:** You can choose which country's currency to use	8 bytes
AutoNumber	AutoNumber fields automatically give each record its own number, usually starting at 1	4 bytes
Date/Time	Use for dates and times	8 bytes
Yes/No	A logical field can be displayed as Yes/No, True/False, or On/Off. In code, use the constants True and False (equivalent to -1 and 0). **Note:** Null values are not allowed in Yes/No fields	1 bit
Ole Object	Can store pictures, audio, video, or other BLOBs (Binary Large Objects)	up to 1GB
Hyperlink	Contain links to other files, including web pages	
Lookup Wizard	Let you type a list of options, which can then be chosen from a drop-down list	4 bytes

(Πηγή: https://www.w3schools.com/sql/sql_datatypes.asp)

4.3. Αναφορές στην SQL (References)

Λέξεις-κλειδιά στην SQL

Λέξη-κλειδί	Περιγραφή
ADD	Προσθέτει μια στήλη σε έναν υπάρχοντα πίνακα
ADD CONSTRAINT	Προσθέτει έναν περιορισμό μετά την δημιουργία ενός πίνακα
ALL	Εμφανίζεται TRUE αν όλες οι τιμές ενός δευτερεύοντος ερωτήματος πληρούν την προϋπόθεση
ALTER	Προσθέτει, διαγράφει ή τροποποιεί στήλες σε έναν πίνακα ή αλλάζει τον τύπο δεδομένων μιας στήλης σε έναν πίνακα
ALTER COLUMN	Αλλάζει τον τύπο δεδομένων μιας στήλης σε έναν πίνακα
ALTER TABLE	Προσθέτει, διαγράφει ή τροποποιεί στήλες σε έναν πίνακα
AND	Περιλαμβάνει μόνο γραμμές όπου και οι δύο συνθήκες είναι αληθείς
ANY	Εμφανίζεται TRUE αν όλες οι τιμές των υποερωτημάτων πληρούν την προϋπόθεση

AS	Μετονομασία στήλης ή πίνακα με ψευδώνυμο
ASC	Ταξινόμηση του αποτελέσματος σε αύξουσα σειρά
BACKUP DATABASE	Δημιουργεί ένα εφεδρικό αντίγραφο μιας υπάρχουσας βάσης δεδομένων
BETWEEN	Επιλογή τιμών εντός ενός δοσμένου εύρους
CASE	Εμφανίζει διαφορετικά αποτελέσματα με βάση τις συνθήκες
CHECK	Ένας περιορισμός που θέτει ένα όριο στην τιμή που μπορεί να τοποθετηθεί σε μια στήλη
COLUMN	Αλλάζει τον τύπο δεδομένων μιας στήλης ή διαγράφει μια στήλη σε έναν πίνακα
CONSTRAINT	Προσθέτει ή διαγράφει έναν περιορισμό
CREATE	Δημιουργεί μια βάση δεδομένων, δείκτη, προβολή, πίνακα ή διαδικασία
CREATE DATABASE	Δημιουργεί μια νέα βάση δεδομένων SQL

CREATE INDEX	Δημιουργεί έναν δείκτη σε έναν πίνακα (επιτρέπει διπλότυπες τιμές)
CREATE OR REPLACE VIEW	Ενημερώνει μια προβολή
CREATE TABLE	Δημιουργεί έναν νέο πίνακα στη βάση δεδομένων
CREATE PROCEDURE	Δημιουργεί μια αποθηκευμένη διαδικασία
CREATE UNIQUE INDEX	Δημιουργεί ένα μοναδικό δείκτη σε έναν πίνακα (χωρίς διπλές τιμές)
CREATE VIEW	Δημιουργεί μια προβολή με βάση το σύνολο αποτελεσμάτων μιας εντολής SELECT
DATABASE	Δημιουργεί ή διαγράφει μια βάση δεδομένων SQL
DEFAULT	Ένας περιορισμός που παρέχει μια προεπιλεγμένη τιμή για μια στήλη
DELETE	Διαγράφει σειρές από έναν πίνακα
DESC	Ταξινόμηση των αποτελεσμάτων κατά φθίνουσα σειρά

DISTINCT	Επιλέγει μόνο διακριτές (διαφορετικές) τιμές
DROP	Διαγράφει μια στήλη, περιορισμό, βάση δεδομένων, δείκτη, πίνακα ή προβολή
DROP COLUMN	Διαγράφει στήλες από έναν πίνακα
DROP CONSTRAINT	Διαγράφει έναν περιορισμό UNIQUE, PRIMARY KEY, FOREIGN KEY, ή CHECK
DROP DATABASE	Διαγράφει μια υπάρχουσα βάση δεδομένων SQL
DROP DEFAULT	Διαγράφει έναν περιορισμό DEFAULT
DROP INDEX	Διαγράφει έναν δείκτη σε έναν πίνακα
DROP TABLE	Διαγράφει έναν υπάρχοντα πίνακα στη βάση δεδομένων
DROP VIEW	Διαγράφει μια προβολή
EXEC	Εκτελεί μια αποθηκευμένη διαδικασία
EXISTS	Εξετάζει την ύπαρξη οποιασδήποτε εγγραφής σε ένα υποερώτημα

FOREIGN KEY	Ένας περιορισμός που αποτελεί το κλειδί που χρησιμοποιείται για τη σύνδεση δύο πινάκων
FROM	Καθορίζει από ποιον πίνακα θα επιλεγούν ή θα διαγραφούν δεδομένα
FULL OUTER JOIN	Εμφανίζει όλες τις σειρές όταν υπάρχει αντιστοιχία είτε στον πίνακα στα αριστερά είτε στα δεξιά
GROUP BY	Ομαδοποιεί το σύνολο αποτελεσμάτων (χρησιμοποιείται με αθροιστικές συναρτήσεις: COUNT, MAX, MIN, SUM, AVG)
HAVING	Χρησιμοποιείται αντί για το WHERE με αθροιστικές συναρτήσεις
IN	Σας επιτρέπει να καθορίσετε πολλαπλές τιμές σε έναν όρο WHERE
INDEX	Δημιουργεί ή διαγράφει έναν δείκτη σε έναν πίνακα
INNER JOIN	Επιλέγει σειρές που έχουν τιμές που είναι ίδιες και στους δύο πίνακες
INSERT INTO	Εισάγει νέες σειρές σε έναν πίνακα
INSERT INTO SELECT	Αντιγράφει δεδομένα από έναν πίνακα σε άλλον πίνακα

IS NULL	Εξετάζει το ενδεχόμενο κενών τιμών
IS NOT NULL	Εξετάζει το ενδεχόμενο μη κενών τιμών
JOIN	Συγχωνεύει πίνακες
LEFT JOIN	Εμφανίζει όλες τις σειρές από τον αριστερό πίνακα και τις σειρές που είναι ίδιες από τον δεξί πίνακα
LIKE	Αναζητά ένα συγκεκριμένο μοτίβο σε μια στήλη
LIMIT	Καθορίζει τον αριθμό των εγγραφών που θα εμφανιστούν στο σύνολο αποτελεσμάτων
NOT	Περιλαμβάνει μόνο σειρές όπου μια συνθήκη δεν είναι TRUE
NOT NULL	Ένας περιορισμός που επιβάλλει σε μια στήλη να μην δέχεται κενές τιμές
OR	Περιλαμβάνει σειρές όπου οποιαδήποτε από τις συνθήκες είναι TRUE
ORDER BY	Ταξινόμηση των αποτελεσμάτων κατά αύξουσα ή φθίνουσα σειρά
OUTER JOIN	Εμφανίζει όλες τις σειρές όπου υπάρχει αντιστοιχία είτε στον αριστερό

	είτε στον δεξί πίνακα
PRIMARY KEY	Ένας περιορισμός που προσδιορίζει ως μοναδική κάθε εγγραφή σε έναν πίνακα βάσης δεδομένων
PROCEDURE	Μια αποθηκευμένη δοκιμασία
RIGHT JOIN	Εμφανίζει όλες τις σειρές από το δεξί πίνακα και τις σειρές που είναι ίδιες με τον αριστερό πίνακα
ROWNUM	Καθορίζει τον αριθμό των εγγραφών που θα εμφανιστούν στο σύνολο αποτελεσμάτων
ΕΠΙΛΟΓΗ	Επιλογή δεδομένων από μια βάση δεδομένων
SELECT DISTINCT	Επιλέγει μόνο διακριτές (διαφορετικές) τιμές
SELECT INTO	Αντιγράφει δεδομένα από έναν πίνακα σε έναν νέο πίνακα
SELECT TOP	Καθορίζει τον αριθμό των εγγραφών που θα εμφανιστούν στο σύνολο αποτελεσμάτων
SET	Καθορίζει ποιες στήλες και τιμές θα πρέπει να ενημερωθούν σε έναν πίνακα

TABLE	Δημιουργεί έναν πίνακα ή προσθέτει, διαγράφει ή τροποποιεί στήλες σε έναν πίνακα ή διαγράφει έναν πίνακα ή τα δεδομένα μέσα σε αυτόν
TOP	Καθορίζει τον αριθμό των εγγραφών που θα εμφανιστούν στο σύνολο αποτελεσμάτων
TRUNCATE TABLE	Διαγράφει τα δεδομένα μέσα σε έναν πίνακα, αλλά όχι τον ίδιο τον πίνακα
UNION	Συνδυάζει το σύνολο των αποτελεσμάτων δύο ή περισσότερων εντολών SELECT (μόνο διακριτές τιμές)
UNION ALL	Συνδυάζει το σύνολο αποτελεσμάτων δύο ή περισσότερων εντολών SELECT (επιτρέπει διπλότυπες τιμές)
UNIQUE	Ένας περιορισμός που διασφαλίζει ότι όλες οι τιμές σε μια στήλη είναι μοναδικές
UPDATE	Ενημερώνει τις υπάρχουσες σειρές σε έναν πίνακα
VALUES	Καθορίζει τις τιμές μιας εντολής INSERT INTO
VIEW	Δημιουργεί, ενημερώνει ή διαγράφει μια προβολή

(Πηγή: https://www.w3schools.com/sql/sql_ref_keywords.asp)

Λειτουργίες στο σύστημα MySQL

Για περισσότερες λεπτομέρειες και μια ολοκληρωμένη λίστα με συγκεκριμένες λειτουργίες που χρησιμοποιούνται στο σύστημα MySQL, οι εκπαιδευόμενοι μπορούν να ανατρέξουν σε αυτόν τον σύνδεσμο.

Λειτουργίες στο σύστημα SQL Server

Για περισσότερες λεπτομέρειες και μια ολοκληρωμένη λίστα με συγκεκριμένες λειτουργίες που χρησιμοποιούνται στο σύστημα SQL Server, οι εκπαιδευόμενοι μπορούν να ανατρέξουν σε αυτόν τον σύνδεσμο.

Λειτουργίες στο σύστημα MS Access

Για περισσότερες λεπτομέρειες και μια ολοκληρωμένη λίστα με συγκεκριμένες λειτουργίες που χρησιμοποιούνται στο σύστημα MS Access, οι εκπαιδευόμενοι μπορούν να ανατρέξουν σε αυτόν τον σύνδεσμο.

Σύνοψη της γλώσσας SQL

Για έναν πλήρη κατάλογο των εντολών της SQL και της αντίστοιχης σύνταξής τους, οι εκπαιδευόμενοι μπορούν να ανατρέξουν σε αυτόν τον σύνδεσμο.

4.4 Παραδείγματα της SQL

Παραδείγματα της SQL

Υπάρχει ένας περιεκτικός κατάλογος παραδειγμάτων στην ιστοσελίδα του W3Schools, τα οποία μπορούν να χρησιμοποιήσουν οι εκπαιδευόμενοι για να μελετήσουν μόνοι τους και να εξασκήσουν περαιτέρω τις δεξιότητές τους στη γλώσσα SQL.

Κουίζ για την SQL

Για τους εκπαιδευόμενους που θέλουν να αξιολογήσουν τις γνώσεις και τις δεξιότητές τους στην SQL, ανατρέξτε σε έναν από τους παρακάτω ιστότοπους:

- W3Schools
- Tutorialspoint

Ασκήσεις στην SQL

Για μια ολοκληρωμένη λίστα των ασκήσεων, οι εκπαιδευόμενοι μπορούν να χρησιμοποιήσουν την ιστοσελίδα του W3Schools για να εξασκήσουν τις δεξιότητές τους στην SQL.