Βάσεις Δεδομένων – 6

SQL – A Relational Database Language – Μέρος Α

Δρ. Χαράλαμπος Καρανίκας

Εισαγωγή στην SQL

Στις προηγούμενες ενότητες μελετήσαμε μια **τυπική γλώσσα** ΒΔ, την **Σχεσιακή Άλγεβρα,** η οποία στηρίζεται πάνω στο **Σχεσιακό Μοντέλο.**

Σε αυτή την ενότητα θα μελετήσουμε μια **πραγματική** γλώσσα βάσεων δεδομένων την **SQL (Structured Query Language)** η οποία στηρίζεται τόσο πάνω στη Σχεσιακή Άλγεβρα όσο και πάνω στον Λογισμό Πλειάδων.

Συγκεκριμένα θα μελετήσουμε την SQL σε δυο **βασικές ενότητες**:

- *Γλώσσα Ορισμού Δεδομένων (Data Definition Language,* ***SQL-DDL****)*
- Γλώσσα Επεξεργασίας *Δεδομένων* (Data Manipulation Language, **SQL-DML**)

Παραδείγματα DDL/DML

Παράδειγμα *Γλώσσας Ορισμού Δεδομένων (Data Definition Language, **SQL-DDL**)*

```
CREATE TABLE DEPARTMENT (
   DNAME              VARCHAR(10) NOT NULL,
   DNUMBER            INTEGER     NOT NULL,
   MGRSSN             CHAR(9),
   MGRSTARTDATE       CHAR(9)
);
```

*Παράδειγμα Γλώσσας Επεξεργασίας Δεδομένων (Data Manipulation Language, **DML**)*

```
SELECT D.MGRSSN, D.MGRSTARTDATE,
FROM DEPARTMENT AS D
WHERE DNUMBER = 10;
```

Ιστορία SQL

1969: *Το Σχεσιακό Μοντέλο του Edgar F. Codd υλοποιείται από τη βάση* ***IBM System R***

1970: *Ο* **Donald D. Chamberlin** και ο **Raymond F. Boyce,** επιστήμονες της IBM Almaden, δημιουργούν την πρώτη έκδοση της SQL η όποια ονομάζεται **SEQUEL**.

- Το όνομα αλλάζει αργότερα σε SQL εφόσον το όνομα SEQUEL ήταν κατοχυρωμένο σε κάποια εταιρεία κατασκευής αεροπλάνων στην Αγγλία.
- Την ίδια χρονιά η Relational Software, Inc., επηρεασμένο από τους Codd, Chamberlin, and Boyce, φτιάχνει την βάση δεδομένων Oracle για κρατικές υπηρεσίες των ΗΠΑ.

1985: *Η* ***IBM*** *κάνει την* ***SQL*** *Πατέντα (US Pat. 4,506,326).*

Οι πιο επιτυχημένες προσπάθειες **τυποποίησης**

- SQL (ANSI 1986)
- SQL1 (ANSI 1989)
- SQL2 ή SQL92 (ANSI 1992)
- **SQL3 ή SQL99 (ANSI 1999) (+OLAP, XML, object-relational etc.)**
- Πρότυπα που έμειναν κυρίως στις συστάσεις: SQL 2003 και SQL 2008

SQL

- Η SQL είναι μια τυπική γλώσσα για την πρόσβαση και τον χειρισμό βάσεων δεδομένων.
- Η SQL δεν είναι ευαίσθητη σε πεζά-κεφαλαία: η select είναι η ίδια με την SELECT.

- Ορισμένες από τις πιο σημαντικές εντολές SQL:

SELECT, UPDATE, DELETE, INSERT INTO, CREATE DATABASE, ALTER DATABASE, CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, DROP INDEX

SQL Τύποι δεδομένων

- Αριθμητικοί
- Σειρά χαρακτήρων
- Σειρά από δυαδικά ψηφία (Bit)
- Χρονολογικά δεδομένα
- NULL τιμή έγκυρη για όλους τους τύπους

char(n). Fixed length character string, with user-specified length *n*.

varchar(n). Variable length character strings, with user-specified maximum length *n*.

int. Integer (a finite subset of the integers that is machine-dependent).

smallint. Small integer (a machine-dependent subset of the integer domain type).

numeric(p,d). Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.

real, double precision. Floating point and double-precision floating point numbers, with machine-dependent precision.

float(n). Floating point number, with user-specified precision of at least *n* digits.

Τύποι δεδομένων MySQL

Text types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	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(x,y,z,etc.)	Let you enter 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. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Τύποι δεδομένων MySQL

Number types:

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

Τύποι δεδομένων MySQL

Date types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*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:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

Ορισμός δεδομένων στην SQL

Μια ***Γλώσσα Ορισμού Δεδομένων (Data Definition Language, SQL-DDL)*** μας επιτρέπει να **δημιουργήσουμε** και να διαχειριστούμε τις δομές μιας βάσης δεδομένων.

Αυτό επιτυγχάνεται μέσω κάποιων εντολών της SQL, π.χ.,:

- **CREATE** δημιουργεί ένα αντικείμενο βάσης
  - π.χ., σχήμα βάσης, πίνακα, ευρετήριο, σκανδάλη, τύπο δεδ., κτλ.
- **DELETE/DROP/TRUNCATE**
  - **DELETE** διαγράφει στοιχεία από το αντικείμενο βάσει συνθήκης (χωρίς να αποδεσμεύεται ο χώρος).
  - **TRUNCATE/DROP:** διαγράφει το αντικείμενο / διαγράφει το αντικείμενο και την περιγραφή του αποδεσμεύοντας και τον χώρο.
- **ALTER** μεταβάλει την δομή ενός αντικειμένου
  - Π.χ., προσθήκη στήλης, επιπλέον περιορισμού, κτλ.

SQL CREATE DATABASE

- Η δήλωση CREATE DATABASE χρησιμοποιείται για τη δημιουργία μιας νέας βάσης δεδομένων SQL.
  - CREATE DATABASE *databasename*;
- Μόλις δημιουργηθεί μια βάση δεδομένων, μπορείτε να την ελέγξετε στη λίστα βάσεων δεδομένων με την ακόλουθη εντολή SQL: SHOW DATABASES;

SQL DROP DATABASE

- Η εντολή DROP DATABASE χρησιμοποιείται για την διαγραφή μιας υπάρχουσας βάσης δεδομένων SQL.

  – DROP DATABASE *databasename*;

- Η διαγραφή μιας βάσης δεδομένων θα έχει ως αποτέλεσμα την απώλεια όλων των πληροφοριών που είναι αποθηκευμένες στη βάση δεδομένων.

CREATE TABLE…

Με χρήση της εντολής CREATE TABLE μπορούμε να ορίσουμε τους ακόλουθους περιορισμούς:

- **Γνωρίσματα**
  - Ορίζεται ο τύπος δεδομένων κάθε γνωρίσματος, π.χ., age int,
- **Πρωτεύοντα** κλειδιά (**PRIMARY KEY**)
  - Ορίζεται το Κλειδί κάθε Σχέσης
  - Δεν μπορεί να είναι NULL
  - Δεν είναι απαραίτητο να υπάρχει εάν και συνίσταται να υπάρχει
- **Δευτερεύοντα** κλειδιά (**UNIQUE**)
  - Ορίζονται τα μοναδικά γνωρίσματα μιας Σχέσης (π.χ., Dname στο Department)
  - Μπορεί να είναι NULL (βασική διαφορά από τα Primary Keys)
- **Κανόνες Αναφορικής Ακεραιότητας** (**FOREIGN KEY**).

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Παράδειγμα … CREATE TABLE

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

- Ο άδειος πίνακας ”Persons" θα φαίνεται ως εξής:

PersonID	LastName	FirstName	Address	City

- Ο άδειος πίνακας μπορεί τώρα να συμπληρωθεί με δεδομένα χρησιμοποιώντας την εντολή SQL INSERT INTO.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SQL DROP, TRUNCATE TABLE

- Η εντολή **DROP TABLE** χρησιμοποιείται για την διαγραφή ενός υπάρχοντος πίνακα σε μια βάση δεδομένων. Η διαγραφή ενός πίνακα θα έχει ως αποτέλεσμα την απώλεια όλων των πληροφοριών που είναι αποθηκευμένες στον πίνακα.

```
DROP TABLE table_name;
```

- Η εντολή **TRUNCATE TABLE** χρησιμοποιείται για τη διαγραφή των δεδομένων μέσα σε έναν πίνακα, αλλά όχι για τον ίδιο τον πίνακα.

```
TRUNCATE TABLE table_name;
```

SQL ALTER TABLE

- Η εντολή ALTER TABLE χρησιμοποιείται για να:
  - προσθέσετε, διαγράψετε ή να τροποποιήσετε στήλες σε έναν υπάρχοντα πίνακα.
  - προσθέσετε διάφορους περιορισμούς σε έναν υπάρχοντα πίνακα.

```
ALTER TABLE table_name
ADD column_name datatype;

ALTER TABLE table_name
DROP COLUMN column_name;

ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Περιορισμοί SQL (SQL Constraints) 1/3

- Οι περιορισμοί SQL χρησιμοποιούνται για τον καθορισμό κανόνων για τα δεδομένα σε έναν πίνακα.
- Μπορούν να καθοριστούν περιορισμοί όταν ο πίνακας δημιουργείται με τη δήλωση CREATE TABLE ή μετά την δημιουργία του πίνακα με την παράμετρο ALTER TABLE.

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Περιορισμοί SQL (SQL Constraints) 2/3

- Οι περιορισμοί χρησιμοποιούνται για τον **περιορισμό του τύπου των δεδομένων** που μπορούν να μεταφερθούν σε έναν πίνακα.
- Αυτό εξασφαλίζει την **ακρίβεια** και την **αξιοπιστία** των δεδομένων στον πίνακα. Εάν υπάρχει παραβίαση μεταξύ του περιορισμού και μιας ενέργειας στα δεδομένα, η ενέργεια διακόπτεται.
- Οι περιορισμοί μπορούν να είναι σε **επίπεδο στήλης** ή σε **επίπεδο πίνακα.** Οι περιορισμοί επιπέδου στήλης ισχύουν για μια στήλη και οι περιορισμοί επιπέδου πίνακα ισχύουν για ολόκληρο τον πίνακα.

Περιορισμοί SQL (SQL Constraints) 3/3

Οι ακόλουθοι περιορισμοί χρησιμοποιούνται συνήθως στην SQL:

- **NOT NULL** - Εξασφαλίζει ότι μια στήλη δεν μπορεί να έχει τιμή NULL
- **UNIQUE** - Διασφαλίζει ότι όλες οι τιμές σε μια στήλη είναι διαφορετικές
- **PRIMARY KEY** - Ένας συνδυασμός ενός NOT NULL και ενός UNIQUE. Μοναδικά αναγνωρίζει κάθε γραμμή σε έναν πίνακα
- **FOREIGN KEY** - Μοναδικά αναγνωρίζει μια σειρά / εγγραφή σε έναν άλλο πίνακα
- **CHECK** - Εξασφαλίζει ότι όλες οι τιμές μιας στήλης ικανοποιούν μια συγκεκριμένη συνθήκη
- **DEFAULT** - Ορίζει μια προεπιλεγμένη τιμή για μια στήλη όταν δεν έχει οριστεί τιμή
- **INDEX** - Χρησιμοποιείται για τη γρήγορη δημιουργία και ανάκτηση δεδομένων από τη βάση δεδομένων

NOT NULL Περιορισμός

- Από προεπιλογή, μια στήλη μπορεί να διατηρήσει τιμές NULL.
- NOT NULL σημαίνει ότι δεν μπορείτε να εισαγάγετε νέα εγγραφή ή να ενημερώσετε μια εγγραφή χωρίς να προσθέσετε μια τιμή σε αυτό το πεδίο.
- Εάν ο πίνακας έχει ήδη δημιουργηθεί, μπορείτε να προσθέσετε έναν περιορισμό NOT NULL σε μια στήλη με την παράμετρο ALTER TABLE.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

UNIQUE Περιορισμός

- Τόσο οι περιορισμοί UNIQUE όσο και PRIMARY KEY αποτελούν εγγύηση για τη μοναδικότητα μιας στήλης ή ενός συνόλου στηλών.
- Ο περιορισμός PRIMARY KEY έχει αυτόματα έναν περιορισμό UNIQUE.
- Ωστόσο, μπορείτε να έχετε πολλούς UNIQUE περιορισμούς ανά πίνακα, αλλά μόνο έναν περιορισμό PRIMARY KEY ανά πίνακα.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,L
astName)
);
```

PRIMARY KEY Περιορισμός

- Ο περιορισμός **PRIMARY KEY** αναγνωρίζει με μοναδικό τρόπο κάθε εγγραφή σε έναν πίνακα βάσης δεδομένων.
- Τα πρωτεύοντα κλειδιά πρέπει να περιέχουν τιμές UNIQUE και δεν μπορούν να περιέχουν τιμές NULL.
- Ένας πίνακας μπορεί να έχει μόνο ένα πρωτεύον κλειδί, το οποίο μπορεί να αποτελείται από ένα ή πολλαπλά πεδία.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,Las
tName)
);
```

FOREIGN KEY Περιορισμός

- Ένα ξένο κλειδί είναι ένα κλειδί που χρησιμοποιείται για τη σύνδεση δύο πινάκων.
- Ένα FOREIGN KEY σε ένα πίνακα δείχνει ένα πρωτεύον κλειδί σε άλλο πίνακα.
- Ο περιορισμός FOREIGN KEY χρησιμοποιείται για την αποτροπή ενεργειών που θα καταστρέψουν τους συνδέσμους μεταξύ των πινάκων.
- Ο περιορισμός FOREIGN KEY εμποδίζει επίσης την εισαγωγή μη έγκυρων δεδομένων στη στήλη ξένου κλειδιού, επειδή πρέπει να είναι μία από τις τιμές που περιέχονται στον πίνακα στον οποίο αναφέρεται.

Παράδειγμα FOREIGN KEY

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

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

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);

ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

CHECK Περιορισμός

- Ο περιορισμός CHECK χρησιμοποιείται για να περιορίσει το εύρος τιμών που μπορεί να τοποθετηθεί σε μια στήλη.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (A
ge>=18 AND City='Sandnes');
```

DEFAULT Περιορισμός

- Ο περιορισμός DEFAULT χρησιμοποιείται για να παρέχει μια προεπιλεγμένη τιμή για μια στήλη.
- Η προεπιλεγμένη τιμή θα προστεθεί σε όλες τις νέες εγγραφές, αν δεν έχει καθοριστεί άλλη τιμή.
- Ο περιορισμός DEFAULT μπορεί επίσης να χρησιμοποιηθεί για την εισαγωγή τιμών συστήματος, χρησιμοποιώντας πχ GETDATE ():

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

```
CREATE TABLE Orders (
    ID int NOT NULL,
    OrderNumber int NOT NULL,
    OrderDate date DEFAULT GETDATE()
);
```

CREATE INDEX

- Τα **ευρετήρια (Indexes)** χρησιμοποιούνται για την γρήγορη ανάκτηση δεδομένων από τη βάση δεδομένων. Οι χρήστες δεν μπορούν να δουν τα ευρετήρια, απλώς χρησιμοποιούνται για την επιτάχυνση των αναζητήσεων / των ερωτημάτων.
- Ένα ευρετήριο για ένα γνώρισμα διατηρεί μια ταξινομημένη ή γενικότερα μια οργανωμένη δομή όλων των τιμών του γνωρίσματος – Όταν ζητηθεί μία τιμή ή ένα πεδίο τιμών για το γνώρισμα, βρίσκουμε τη θέση των τιμών αυτών μέσα στο ευρετήριο και από εκεί βρίσκουμε κατευθείαν τις γραμμές του πίνακα που ζητάμε.

Επιτρέπονται διπλές τιμές

```
CREATE INDEX index_name
ON table_name (column1, column2,
...);
```

Δεν επιτρέπονται διπλές τιμές

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Επεξεργασία δεδομένων με SQL

SQL Data Manipulation Language

SELECT Statement

- Η εντολή SELECT χρησιμοποιείται για την επιλογή δεδομένων από μια βάση δεδομένων.
- Τα δεδομένα που επιστρέφονται αποθηκεύονται σε έναν πίνακα αποτελεσμάτων.

```
SELECT column1, column2, ...
FROM table_name;

SELECT * FROM table_name;
```

- Η εντολή SELECT DISTINCT χρησιμοποιείται για την επιστροφή μόνο διαφορετικών τιμών.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

WHERE Clause

- Χρησιμοποιείται για να φιλτράρει τις εγγραφές.
- Το WHERE δεν χρησιμοποιείται μόνο στην εντολή SELECT, αλλά χρησιμοποιείται και στην εντολή UPDATE, DELETE κ.λπ.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

WHERE Clause – Operators 1/2

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

WHERE Clause – Operators 2/2

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

ORDER BY

- Το ORDER BY χρησιμοποιείται για να ταξινομήσει το πίνακα αποτελεσμάτων σε αύξουσα ή φθίνουσα σειρά.
- Ταξινομεί σε αύξουσα σειρά από προεπιλογή. Για να ταξινομήσετε σε φθίνουσα σειρά, χρησιμοποιήστε τη λέξη-κλειδί DESC.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

INSERT INTO Statement

- Η εντολή INSERT INTO χρησιμοποιείται για την εισαγωγή νέων εγγραφών σε έναν πίνακα.
- Είναι δυνατό να γράψετε την εντολή INSERT INTO με δύο τρόπους:
  - Ο πρώτος τρόπος καθορίζει τόσο τα ονόματα των στηλών όσο και τις τιμές που θα εισαχθούν

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

  - Εάν προσθέτετε τιμές για όλες τις στήλες του πίνακα, δεν χρειάζεται να καθορίσετε τα ονόματα στηλών στο ερώτημα SQL. Ωστόσο, βεβαιωθείτε ότι η σειρά των τιμών είναι στην ίδια σειρά με τις στήλες στον πίνακα.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SQL NULL Values 1/2

- Ένα πεδίο με τιμή NULL είναι ένα πεδίο χωρίς τιμή.
- Εάν ένα πεδίο στον πίνακα είναι προαιρετικό, είναι δυνατό να εισαγάγετε μια νέα εγγραφή ή να ενημερώσετε μια εγγραφή χωρίς να προσθέσετε μια τιμή σε αυτό το πεδίο. Στη συνέχεια, το πεδίο θα αποθηκευτεί με τιμή NULL.
- Είναι πολύ σημαντικό να κατανοήσουμε ότι μια τιμή NULL είναι διαφορετική από μια μηδενική τιμή ή ένα πεδίο που περιέχει κενά διαστήματα. Ένα πεδίο με τιμή NULL είναι ένα πεδίο που έχει μείνει κενό κατά τη δημιουργία εγγραφών.

SQL NULL Values 2/2

Πώς να ελέγξετε τις τιμές NULL;

- Δεν είναι δυνατή η δοκιμή για τιμές NULL με τελεστές σύγκρισης, όπως =, <, ή <>.
- Θα πρέπει να χρησιμοποιήσουμε τους IS NULL και IS NOT NULL operators.

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```