ΕΙΣΑΓΩΓΗ ΣΤΙΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ

Δεύτερη Ενότητα

SQL

Ενημερώσεις στην SQL

Υπάρχουν 3 εντολές στην SQL για τροποποίηση της βάσης δεδομένων :

- **INSERT :** Χρησιμοποιείται για εισαγωγή πλειάδων σε μια σχέση
- **DELETE :** Διαγράφει πλειάδες από μια σχέση
- **UPDATE :** Τροποποιεί τιμές γνωρισμάτων μιας ή περισσότερων πλειάδων

SQL

Η εντολή INSERT

Στην απλούστερη μορφή της, χρησιμοποιείται για εισαγωγή πλειάδων σε μια σχέση. Οι τιμές των γνωρισμάτων πρέπει να δοθούν με την ίδια σειρά που ορίσθηκαν στην εντολή CREATE TABLE.

Παράδειγμα:

INSERT INTO Persons
VALUES (‘Ευσταθίου', ‘Νίκος‘ , ‘Λυκούργου 10', ‘Νέα Ιωνία')

Μια εναλλακτική μορφή της INSERT ορίζει ρητά τα ονόματα των γνωρισμάτων που αντιστοιχούν στις τιμές στη νέα πλειάδα.
Μπορούν να παραληφθούν γνωρίσματα με τιμή NULL

Παράδειγμα: Εισαγωγή μιας νέας πλειάδα στη σχέση Persons που ξέρουμε μόνο τα γνωρίσματα Lastname, Firstname, και City.

INSERT INTO Persons (Lastname, Firstname, City)
VALUES (‘Σαμαράς', ‘Γεώργιος', ‘Αθήνα')

SQL

Η εντολή DELETE

- Διαγράφει πλειάδες από μια σχέση
- Έχει μια πρόταση WHERE- για επιλογή των πλειάδων που θα διαγραφούν
- Πρέπει να επιβληθεί η αναφορική ακεραιότητα
- Οι πλειάδες διαγράφονται μόνο από ένα πίνακα τη φορά (εκτός αν έχει ορισθεί αναφορικός περιορισμός ακεραιότητας CASCADE)
- Ελλειψη της πρότασης WHERE- ορίζει ότι όλες οι πλειάδες της σχέσης θα διαγραφούν· ο πίνακας που προκύπτει είναι κενός
- Το πλήθος των πλειάδων που διαγράφονται εξαρτάται από το πλήθος των πλειάδων της σχέσης που ικανοποιούν την πρόταση WHERE

Παραδείγματα:

DELETE FROM Persons WHERE Lastname=‘Ευσταθίου’

DELETE FROM Persons WHERE City=‘ΦΛΩΡΙΝΑ’

SQL

Η εντολή UPDATE

- Χρησιμοποιείται για τροποποίηση τιμών γνωρισμάτων μιας ή περισσότερων πλειάδων
- Μια πρόταση WHERE επιλέγει τις πλειάδες που θα τροποποιηθούν
- Μια επιπλέον πρόταση SET καθορίζει τα γνωρίσματα που θα τροποποιηθούν και τις νέες τιμές τους
- Κάθε εντολή τροποποιεί πλειάδες στην ίδια σχέση
- Πρέπει να επιβληθούν οι αναφορικοί περιορισμοί ακεραιότητας

Παράδειγμα: Να αλλάξει η η διεύθυνση (Ελ Βενιζέλου 118) και η πόλη (Πάτρα) του Μαρκόπουλου από τον πίνακα Persons

```
UPDATE  Persons
SET Address = 'Ελ Βενιζέλου 118',  City = ‘Πάτρα’
WHERE Lastname= ‘Μαρκόπουλος’
```

SQL

ADVANCED

TOP Clause

Χρησιμοποιείται για να ορίσουμε τον αριθμό των εγγραφών που θα επιστρέψει. Είναι πολύ χρήσιμο σε πίνακες που περιέχουν χιλιάδες εγγραφές , καθώς τα πολλά αποτελέσματα που μπορεί να επιστρέψει ένα ερώτημα μπορεί να έχουν αντίκτυπο στην απόδοση.

Προσοχή : Στη MySQL χρησιμοποιείται με διαφορετικό τρόπο.

Παράδειγμα σύνταξης :

```
SELECT TOP 2 * FROM Persons
```

Θα επιστρέψει τις 2 πρώτες εγγραφές

Παράδειγμα σύνταξης MySQL :

```
SELECT  * FROM Persons LIMIT 2
```

Θα επιστρέψει τις 2 πρώτες εγγραφές

SQL

ADVANCED

Wildcards

Τα SQL wildcards μπορούν να χρησιμοποιηθούν κατά την αναζήτηση δεδομένων σε μια βάση δεδομένων. Τα wildcards πρέπει να χρησιμοποιούνται με τον SQL τελεστή LIKE.

Wildcard	Περιγραφή
%	Ένα υποκατάστατο του 0 και άλλων χαρακτήρων
_	Ένα υποκατάστατο για ακριβώς ένα χαρακτήρα
[charlist]	Κάθε μεμονωμένο χαρακτήρα σε λίστα χαρακτήρων
[^charlist] or [!charlist]	Κάθε μεμονωμένο χαρακτήρα που δεν περιλαμβάνεται στη λίστα χαρακτήρων

Persons

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

Παραδείγματα εντολών για τον πίνακα Persons χρησιμοποιώντας το wildcard % :

1) SELECT * FROM Persons WHERE City LIKE 'sa%‘
2) SELECT * FROM Persons WHERE City LIKE '%nes%‘

SQL

ADVANCED

Wildcards

Απαντήσεις

1)

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

2)

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

Παραδείγματα εντολών για τον πίνακα Persons χρησιμοποιώντας το wildcard _ :

1) SELECT * FROM Persons WHERE FirstName LIKE '_la‘
2) SELECT * FROM Persons WHERE LastName LIKE 'S_end_on'

Απαντήσεις

1)

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes

2)

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

SQL

ADVANCED

Wildcards

Παραδείγματα εντολών για τον πίνακα Persons χρησιμοποιώντας το wildcard [charlist] :

1) SELECT * FROM Persons WHERE LastName LIKE '[bsp]%‘
2) SELECT * FROM Persons WHERE LastName LIKE '[!bsp]%'

Απαντήσεις

1)

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

2)

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes

SQL

ADVANCED

In

Ο τελεστής in σας επιτρέπει να καθορίσετε πολλές τιμές σε ένα WHERE clause.

Σύνταξη :

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...)
```

Παραδείγματα εντολών για τον πίνακα Persons χρησιμοποιώντας τον τελεστή in :

1) SELECT * FROM Persons WHERE LastName IN ('Hansen','Pettersen')

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

SQL

ADVANCED

Alias

Με την SQL, ένα Alias(ψευδώνυμο) μπορεί να δοθεί σε ένα πίνακα ή σε μια στήλη. Αυτό μπορεί να είναι καλό όταν έχετε να κάνετε με πολύ μεγάλα ή πολύπλοκα ονόματα πινάκων ή ονόματα στηλών. Ένα ψευδώνυμο θα μπορούσε να είναι οτιδήποτε, αλλά συνήθως είναι μικρό σε μέγεθος και προσδιορίζει ακριβώς αυτό που θέλουμε.

Σύνταξη για πίνακες :

```
SELECT column_name(s)
FROM table_name
AS alias_name
```

Σύνταξη για στήλες :

```
SELECT column_name AS alias_name
FROM table_name
```

Ας υποθέσουμε ότι έχουμε έναν πίνακα που ονομάζεται “Persons" και ένα άλλο πίνακα που ονομάζεται "Product_Orders". Θα δώσουμε τα ψευδώνυμα πίνακα του “p" και “po" αντίστοιχα.

Τώρα θέλουμε να απαριθμήσουμε όλες τις παραγγελίες για τις οποίες είναι υπεύθυνη η "Ola Hansen" .
Χρησιμοποιούμε την ακόλουθη δήλωση SELECT:

```
SELECT po.OrderID, p.LastName, p.FirstName
FROM Persons AS p,
Product_Orders AS po
WHERE p.LastName='Hansen' AND p.FirstName='Ola
```

Η ίδια δήλωση SELECT χωρίς aliases:

```
SELECT Product_Orders.OrderID, Persons.LastName,
Persons.FirstName
FROM Persons,
Product_Orders
WHERE Persons.LastName='Hansen' AND
Persons.FirstName='Ola'
```

Όπως θα δείτε από τις δύο παραπάνω προτάσεις SELECT τα aliases μπορούν να κάνουν ερωτήματα εύκολο να τα γράψει κανείς καθώς και να τα διαβάσει.

ADVANCED

Joins

Τα SQL joins χρησιμοποιούνται για την αναζήτηση δεδομένων από δύο ή περισσότερους πίνακες, βασισμένα σε μια σχέση μεταξύ ορισμένων στηλών αυτών των πινάκων.

Οι πίνακες σε μια βάση δεδομένων συχνά συνδέονται μεταξύ τους με κλειδιά.

Ένα πρωτεύον κλειδί είναι μια στήλη (ή ένας συνδυασμός στηλών) με μια μοναδική τιμή για κάθε γραμμή. Κάθε τιμή του πρωτεύοντος κλειδιού πρέπει να είναι μοναδική μέσα στον πίνακα. Ο σκοπός είναι να συνδεθούν τα δεδομένα μαζί, σε πίνακες, χωρίς επανάληψη όλων των δεδομένων σε κάθε πίνακα.

Έχουμε τους 2 παρακάτω πίνακες Persons και Orders όπου τους συνδέει το P_ID :

Persons

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

Orders

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

SQL

ADVANCED

Joins

Θα δούμε μια λίστα με τα είδη των JOIN που μπορούμε να χρησιμοποιήσουμε, και τις διαφορές μεταξύ τους.

JOIN: Επιστρέφει τις γραμμές όταν υπάρχει τουλάχιστον μια αντιστοιχία και στους δύο πίνακες
LEFT JOIN: Επιστρέφει όλες τις γραμμές από τον αριστερό πίνακα, έστω και αν δεν υπάρχουν αντιστοιχίες στο δεξιό πίνακα
RIGHT JOIN: Επιστρέφει όλες τις γραμμές από τον δεξιό πίνακα , ακόμη και αν δεν υπάρχουν αντιστοιχίες στο αριστερό πίνακα
FULL JOIN: Επιστρέφει τις γραμμές όταν υπάρχει μια αντιστοιχία και στους δύο πίνακες

Inner Join

Σύνταξη :

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON
table_name1.column_name=table_name2.column_nam
e
```

ΥΓ. Το INNER JOIN είναι το ίδιο με το JOIN

SQL

ADVANCED

Inner Join

Με βάση τους πίνακες Persons και Orders που είδαμε παραπάνω , θέλουμε να εμφανίσουμε όλα τα άτομα που πραγματοποίησαν παραγγελίες

Σύνταξη :

```
SELECT Persons.LastName, Persons.FirstName,
Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

Το αποτέλεσμα :

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

Το INNER JOIN επιστρέφει τις γραμμές όταν υπάρχει τουλάχιστον μια αντιστοιχία στους δύο πίνακες. Εάν υπάρχουν γραμμές στον πίνακα Persons που δεν έχουν αντιστοιχία στον πίνακα Orders, οι γραμμές αυτές δεν θα εμφανισθούν.

SQL

ADVANCED

Left Join

Επιστρέφει όλες τις γραμμές από τον αριστερό πίνακα, έστω και αν δεν υπάρχουν αντιστοιχίες στο δεξιό πίνακα.

Σύνταξη :

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON
table_name1.column_name=table_name2.column_name
```

ΥΓ. Το LEFT JOIN σε κάποιες βάσεις λειτουργεί ως LEFT OUTER JOIN

Με βάση τους πίνακες Persons και Orders που είδαμε παραπάνω , θέλουμε να εμφανίσουμε όλα τα άτομα που πραγματοποίησαν παραγγελίες

Σύνταξη :

```
SELECT Persons.LastName, Persons.FirstName,
Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

Το αποτέλεσμα :

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	

Το LEFT JOIN επιστρέφει όλες τις γραμμές του αριστερού πίνακα (Persons) ακόμη και αν δεν υπάρχει αντιστοιχία στο δεξί πίνακα (Orders).

SQL

ADVANCED

Right Join

Επιστρέφει όλες τις γραμμές από τον δεξιό πίνακα, έστω και αν δεν υπάρχουν αντιστοιχίες στον αριστερό πίνακα.

Σύνταξη :

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON
table_name1.column_name=table_name2.column_name
```

ΥΓ. Το RIGHT JOIN σε κάποιες βάσεις λειτουργεί ως RIGHT OUTER JOIN

Με βάση τους πίνακες Persons και Orders που είδαμε παραπάνω , θέλουμε να εμφανίσουμε όλα τα άτομα που πραγματοποίησαν παραγγελίες

Σύνταξη :

```
SELECT Persons.LastName, Persons.FirstName,
Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

Το αποτέλεσμα :

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
		34764

Το RIGHT JOIN επιστρέφει όλες τις γραμμές του δεξιού πίνακα (Orders) ακόμη και αν δεν υπάρχει αντιστοιχία στον αριστερό πίνακα (Persons).

SQL

ADVANCED

Full Join

Επιστρέφει τις γραμμές όταν υπάρχει μια αντιστοιχία και στους δύο πίνακες

Σύνταξη :

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON
table_name1.column_name=table_name2.column_name
```

Με βάση τους πίνακες Persons και Orders που είδαμε παραπάνω , θέλουμε να εμφανίσουμε όλα τα άτομα και τις Παραγγελίες που πραγματοποίησαν :

Σύνταξη :

```
SELECT Persons.LastName, Persons.FirstName,
Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

Το αποτέλεσμα :

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	
		34764

Το RIGHT JOIN επιστρέφει όλες τις γραμμές του αριστερού πίνακα (Persons) και όλες τις γραμμές του δεξιού πίνακα (Orders) . Αν υπάρχουν γραμμές στον πίνακα Persons και δεν υπάρχει αντιστοιχία στον πίνακα Orders ή ισχύει το αντίθετο , τότε αυτές οι γραμμές θα εμφανισθούν επίσης.

SQL

ADVANCED

Union

Ο τελεστής Union συνδυάζει δυο ή περισσότερες εντολές SELECT.
Κάθε εντολή SELECT μέσα στο Union πρέπει να έχει τον ίδιο αριθμό στηλών. Οι στήλες πρέπει να έχουν επίσης παρόμοιους τύπους δεδομένων. Επίσης, οι στήλες σε κάθε εντολή SELECT πρέπει να είναι με την ίδια σειρά.

ΥΓ. Το UNION δεν επιτρέπει διπλότυπα , αν επιθυμούμε διπλότυπα θα χρησιμοποιήσουμε το UNION ALL

Σύνταξη UNION :

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2
```

Σύνταξη UNION ALL :

```
SELECT column_name(s) FROM table_name1
UNION ALL
SELECT column_name(s) FROM table_name2
```

Θα χρησιμοποιήσουμε τους παρακάτω πίνακες για να δούμε πως λειτουργεί ο τελεστής Union :

Employees_Norway

E_ID	E_Name
01	Hansen, Ola
02	Svendson, Tove
03	Svendson, Stephen
04	Pettersen, Kari

Employees_USA

E_ID	E_Name
01	Turner, Sally
02	Kent, Clark
03	Svendson, Stephen
04	Scott, Stephen

SQL

ADVANCED

Union

Θέλουμε να εμφανίσουμε όλους τους διαφορετικούς υπαλλήλους της Νορβηγίας και της Αμερικής με βάση τους Παραπάνω πίνακες :

Σύνταξη :

```
SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA
```

Το αποτέλεσμα :

E_Name
Hansen, Ola
Svendson, Tove
Svendson, Stephen
Pettersen, Kari
Turner, Sally
Kent, Clark
Scott, Stephen

Όπως παρατηρείτε ένα όνομα που υπάρχει και στους δυο πίνακες εμφανίζεται μόνο μια φορά. Για να εμφανιστούν τα διπλότυπα είπαμε πιο πάνω ότι χρησιμοποιούμε το UNION ALL.

SQL

ADVANCED

SELECT INTO

Η εντολή Select into χρησιμοποιείται συνήθως για να δημιουργήσει αντίγραφα των πινάκων. Επιλέγει τα στοιχεία ενός πίνακα και τα εισάγει σε έναν άλλο.

Σύνταξη:

```
SELECT *
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

* Όλες οι στήλες θα αντιγραφούν στο νέο πίνακα

Παράδειγμα για backup:

```
SELECT *
INTO Persons_Backup
FROM Persons
```

Σύνταξη :

```
SELECT column_name(s)
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

* Επιλέγουμε τις στήλες που θα αντιγραφούν στο νέο πίνακα

Παράδειγμα αντιγραφής πίνακα σε εξωτερική βάση:

```
SELECT *
INTO Persons_Backup IN  eksoteriki_vasi.mdb
FROM Persons
```

SQL

ADVANCED

CREATE DATABASE

Η εντολή Create database χρησιμοποιείται για να δημιουργήσει μια νέα βάση δεδομένων :

Σύνταξη:

```
create database όνομα_βάσης
```

Παράδειγμα:

```
create database paradeigma
```

* Όπου paradeigma η βάση δεδομένων μας.

CREATE TABLE

Η εντολή Create database χρησιμοποιείται για να δημιουργήσει ένα νέο πίνακα :

Σύνταξη:

```
create table όνομα_πίνακα
(
όνομα_στήλης1 τύπος_δεδομένων,
όνομα_στήλης2 τύπος_δεδομένων,
όνομα_στήλης3 τύπος_δεδομένων,
....
)
```

Παράδειγμα:

```
create table pinakas1
(
κωδικός int(10),
όνομα varchar(50),
επώνυμο varchar(50)
)
```

Περιορισμοί-Constraints

Οι περιορισμοί χρησιμοποιούνται για να περιορίσουν το είδος των δεδομένων που μπορούν να εισαχθούν σε ένα πίνακα.

Οι Περιορισμοί καθορίζονται, όταν δημιουργείται ένας πίνακας (με τη δήλωση CREATE TABLE) ή μετά τη δημιουργία του πίνακα (με την πρόταση ALTER TABLE).

Ισχύουν οι εξής περιορισμοί:

- **NOT NULL**
- **UNIQUE**
- **PRIMARY KEY**
- **FOREIGN KEY**
- **CHECK**
- **DEFAULT**

SQL

ADVANCED

NOT NULL

Ο περιορισμός NOT NULL επιβάλλει σε μια στήλη να μην αποδεχθεί τις κενές τιμές(NULL).
Επιβάλλει δηλαδή ένα πεδίο να περιέχει πάντα μια τιμή. Αυτό σημαίνει ότι δεν μπορούμε να εισάγουμε μια νέα εγγραφή, ή να ενημερώσουμε ένα αρχείο χωρίς να προσθέσουμε μια τιμή σε αυτό το πεδίο.

Η ακόλουθη SQL επιβάλλει στην στήλη "P_Id" και “LastName" να μην αποδεχθούν κενές τιμές NULL:

Παράδειγμα:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

SQL

ADVANCED

UNIQUE

Η UNIQUE περιορισμός προσδιορίζει μοναδικά κάθε εγγραφή σε έναν πίνακα βάσης δεδομένων.

Η UNIQUE και το PRIMARY KEY περιορισμοί παρέχουν την εγγύηση για την μοναδικότητα μιας στήλη ή ενός σύνολου στηλών.

Το PRIMARY KEY έχει αυτόματα ένα UNIQUE περιορισμό ορισμένο.

*Μπορούμε να έχουμε πολλούς UNIQUE περιορισμούς σε ένα πίνακα, αλλά μόνο ένα PRIMARY KEY.

SQL UNIQUE Constraint on CREATE TABLE

Η ακόλουθη SQL δημιουργεί ένα UNIQUE περιορισμό για τη στήλη "P_Id" όταν ο πίνακας “Persons" δημιουργείται:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
UNIQUE (P_Id)
)
```

SQL

ADVANCED

PRIMARY KEY

Το PRIMARY KEY προσδιορίζει μοναδικά κάθε εγγραφή σε έναν πίνακα βάσης δεδομένων.

Το PRIMARY KEY πρέπει να περιέχει μοναδικές τιμές.

Μια στήλη πρωτεύοντος κλειδιού δεν μπορεί να περιέχει τιμές NULL.

Κάθε πίνακας πρέπει να έχει ένα πρωτεύον κλειδί και κάθε πίνακας μπορεί να έχει μόνο ένα πρωτεύον κλειδί.

Η ακόλουθη εντολή δημιουργεί ένα πρωτεύον κλειδί για τη στήλη "P_Id" όταν ο πίνακας “Persons" δημιουργείται:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

SQL

ADVANCED

FOREIGN KEY

Το FOREIGN KEY δείχνει σε ένα άλλο PRIMARY KEY σε έναν άλλο πίνακα.

Έχουμε τους δυο παρακάτω πίνακες :

Persons:

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

Orders :

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Το P_Id του πίνακα Persons δείχνει στο P_Id στον πίνακα Orders.

Το P_Id του πίνακα Persons είναι το PRIMARY KEY του πίνακα Persons.

Το P_Id του πίνακα Orders είναι το FOREIGN KEY του πίνακα Orders.

SQL

ADVANCED

FOREIGN KEY

Το FOREIGN KEY χρησιμοποιείται για την πρόληψη ενεργειών που θα μπορούσαν να καταστρέψουν σχέσεις μεταξύ πινάκων.
Το FOREIGN KEY εμποδίζει επίσης μη έγκυρα δεδομένα να εισαχθούν στη στήλη του ξένου κλειδιού, επειδή θα πρέπει να είναι μία από τις τιμές που περιέχονται στον πίνακα που αυτό δείχνει.

Η ακόλουθη SQL δημιουργεί ένα FOREIGN KEY για τη στήλη "P_Id" όταν ο πίνακας “Orders" δημιουργείται:

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
)
```

CHECK CONSTRAINT-Περιορισμός Ελέγχου

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

Αν ορίσετε έναν περιορισμό ελέγχου σε μία μόνο στήλη επιτρέπει μόνο ορισμένες τιμές για αυτή τη στήλη.

Αν ορίσετε έναν περιορισμό ελέγχου σε ένα πίνακα μπορεί να περιορίσει τις τιμές σε ορισμένες στήλες με βάση τις τιμές σε άλλες στήλες στη σειρά.

Η ακόλουθη SQL δημιουργεί έναν περιορισμό ελέγχου για τη στήλη "P_Id" όταν ο πίνακας “Persons" δημιουργείται. Ο περιορισμός CHECK διευκρινίζει ότι η στήλη "P_Id" πρέπει να περιλαμβάνει μόνο ακέραιους μεγαλύτερους από 0.

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P_Id>0)
)
```

DEFAULT CONSTRAINT-Περιορισμός Ελέγχου

Ο περιορισμός DEFAULT χρησιμοποιείται για την εισαγωγή μιας προεπιλεγμένης τιμή σε μια στήλη.

Η προεπιλεγμένη τιμή θα προστεθεί σε όλες τις νέες εγγραφές, αν δεν έχει καθοριστεί μια άλλη τιμή.

Η ακόλουθη SQL δημιουργεί έναν προεπιλεγμένο περιορισμό για το πεδίο "City", όταν ο πίνακας "Persons" δημιουργήθηκε:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'Sandnes'
)
```

SQL

ADVANCED

INDEXES-Ευρετήρια

Η πρόταση CREATE INDEX χρησιμοποιείται για τη δημιουργία ευρετηρίων σε πίνακες.

Τα ευρετήρια επιτρέπουν στην εφαρμογή που διαχειρίζεται τη βάση δεδομένων να βρεί τα δεδομένα γρήγορα , χωρίς να διαβάσει ολόκληρο τον πίνακα.

Οι χρήστες δεν μπορούν να δουν τα ευρετήρια, απλώς χρησιμοποιούνται για να επιταχύνουν τις αναζητήσεις και τα ερωτήματα.

Σημείωση: Η ενημέρωση ενός πίνακα με ευρετήρια παίρνει περισσότερο χρόνο από ό, τι η ενημέρωση ενός πίνακα χωρίς (επειδή τα ευρετήρια πρέπει επίσης να ενημερωθούν). Έτσι, θα πρέπει να δημιουργήσετε ευρετήρια μόνο σε στήλες (και πίνακες) που θα αναζητούνται πληροφορίες συχνά.

Σύνταξη :

```
CREATE INDEX index_name
ON table_name (column_name)
```

Παράδειγμα :

```
CREATE INDEX otidipote
ON Persons (Firstname)
```

Σύνταξη διαγραφής ενός index :

```
ALTER TABLE table_name DROP INDEX index_name
```

Παράδειγμα διαγραφής index :

```
ALTER TABLE Persons
DROP INDEX otidipote
```

SQL

ADVANCED

DROP

DROP TABLE

Η πρόταση DROP TABLE χρησιμοποιείται για τη διαγραφή ενός πίνακα.

Σύνταξη :

```
DROP TABLE table_name
```

Παράδειγμα :

```
DROP TABLE Persons
```

TRUNCATE TABLE

Η πρόταση TRUNCATE TABLE χρησιμοποιείται για τη διαγραφή όλων των δεδομένων ενός πίνακα.

Σύνταξη :

```
TRUNCATE TABLE table_name
```

Παράδειγμα :

```
TRUNCATE TABLE Persons
```

DROP DATABASE

Η πρόταση DROP DATABASE χρησιμοποιείται για τη διαγραφή μιας βάσης.

Σύνταξη :

```
DROP DATABASE  database_name
```

Παράδειγμα :

```
DROP DATABASE  videoclub
```

SQL

ADVANCED

ALTER TABLE

Η πρόταση ALTER TABLE χρησιμοποιείται για να προσθέσει , διαγράψει ή να τροποποιήσει τις στήλες ενός υπάρχον πίνακα.

Σύνταξη πρόσθεσης στήλης :

```
ALTER TABLE table_name
ADD column_name datatype
```

Παράδειγμα :

```
ALTER TABLE Persons
ADD dieuthinsi VARCHAR(40)
```

Σύνταξη διαγραφής στήλης :

```
ALTER TABLE table_name
DROP COLUMN column_name
```

Παράδειγμα :

```
ALTER TABLE Persons
DROP COLUMN dieuthinsi
```

Σύνταξη τροποποίησης στήλης :

```
ALTER TABLE table_name
MODIFY column_name datatype
```

Παράδειγμα :

```
ALTER TABLE Persons
MODIFY dieuthinsi TEXT(50)
```

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Στη MySQL που χρησιμοποιούμε εμείς , υπάρχουν τρεις τύποι δεδομένων. Δεδομένα αλφαριθμητικά , αριθμητικά και ημερομηνίας/ώρας.

Αλφαριθμητικά δεδομένα

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ	ΠΕΡΙΓΡΑΦΗ
CHAR(size)	Κρατά ένα σταθερό μέγεθος συμβολοσειράς (Μπορεί να περιέχει γράμματα , αριθμούς ή ειδικούς χαρακτήρες). Το σταθερό μέγεθος προσδιορίζεται μέσα στην παρένθεση. Μπορούν να αποθηκευτούν μέχρι 255 χαρακτήρες
VARCHAR(size)	Κρατά ένα μεταβλητό μέγεθος συμβολοσειράς (Μπορεί να περιέχει γράμματα , αριθμούς ή ειδικούς χαρακτήρες). Το μεταβλητό μέγεθος προσδιορίζεται μέσα στην παρένθεση. Μπορούν να αποθηκευτούν μέχρι 255 χαρακτήρες **Προσοχή:** Αν πληκτρολογήσουμε μέγεθος πάνω από 255 , τότε θα μετατραπεί αυτόματα σε TEXT.
TINYTEXT	Κρατά μια συμβολοσειρά με μέγιστο μέγεθος 255 χαρακτήρες.
TEXT	Κρατά μια συμβολοσειρά με μέγιστο μέγεθος 65,535 χαρακτήρες.
BLOB	Για BLOBs (Binary Large OBjects). Κρατά μέχρι 65,535 bytes δεδομένων.
MEDIUMTEXT	Κρατά μια συμβολοσειρά με μέγιστο μέγεθος 16,777,215 χαρακτήρες.
MEDIUMBLOB	Για BLOBs (Binary Large OBjects). Κρατά μέχρι 16,777,215 bytes δεδομένων.
LONGTEXT	Κρατά μια συμβολοσειρά με μέγιστο μέγεθος 4,294,967,295 χαρακτήρες.
LONGBLOB	Για BLOBs (Binary Large OBjects). Κρατά μέχρι 4,294,967,295 bytes δεδομένων.
ENUM(x,y,z,etc.)	Μπορούμε να δώσουμε μια λίστα με πιθανές τιμές. Μπορούμε να τοποθετήσουμε μέχρι 65,535 τιμές στη λίστα. Αν εισαχθεί μια τιμή που δεν υπάρχει στη λίστα , τότε μια κενή τιμή θα εισαχθεί.
SET	Μοιάζει με το ENUM εκτός του ότι μπορεί να περιέχει μέχρι 64 τιμές και μπορούν να αποθηκευτούν πάνω από μια τιμές.

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Αριθμητικά δεδομένα

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ	ΠΕΡΙΓΡΑΦΗ
TINYINT(size)	-128 μέχρι 127 κανονικά. 0 μέχρι 255 UNSIGNED*.
SMALLINT(size)	-32768 μέχρι 32767 κανονικά. 0 μέχρι 65535 UNSIGNED*.
MEDIUMINT(size)	-8388608 μέχρι 8388607 κανονικά. 0 μέχρι 16777215 UNSIGNED*.
INT(size)	-2147483648 μέχρι 2147483647 κανονικά. 0 μέχρι 4294967295 UNSIGNED*.
BIGINT(size)	-9223372036854775808 μέχρι 9223372036854775807 κανονικά. 0 μέχρι 18446744073709551615 UNSIGNED*.
FLOAT(size,d)	Ένας μικρός αριθμός με μεταβλητό δεκαδικό σημείο. Ο μέγιστος αριθμός ψηφίων ορίζεται όπου το size στην παρένθεση. Ο μέγιστος αριθμός δεκαδικών ψηφίων ορίζεται όπου το d στην παρένθεση. π.χ. FLOAT(3,5) ο αριθμός 999,12345
DOUBLE(size,d)	Ένας μεγάλος αριθμός με μεταβλητό δεκαδικό σημείο. Ο μέγιστος αριθμός ψηφίων ορίζεται όπου το size στην παρένθεση. Ο μέγιστος αριθμός δεκαδικών ψηφίων ορίζεται όπου το d στην παρένθεση.
DECIMAL(size,d)	Ένας DOUBLE αριθμός που αποθηκεύεται ως συμβολοσειρά και επιτρέπει ένα σταθερό μέγεθος δεκαδικού σημείου. Ο μέγιστος αριθμός ψηφίων ορίζεται όπου το size στην παρένθεση. Ο μέγιστος αριθμός δεκαδικών ψηφίων ορίζεται όπου το d στην παρένθεση.

* Οι ακέραιοι τύποι δεδομένων έχουν μια επιπλέον επιλογή που ονομάζεται UNSIGNED. Κανονικά, ο ακέραιος ξεκινά από μια αρνητική και φθάνει σε μια θετική τιμή. Η προσθήκη του UNSIGNED χαρακτηριστικού θα μετακινήσει το εύρος έτσι ώστε να ξεκινά από το μηδέν αντί ενός αρνητικού αριθμού.

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Ημερομηνίας/ώρας δεδομένα

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ	ΠΕΡΙΓΡΑΦΗ
DATE()	Μορφή : YYYY-MM-DD **Προσοχή:** Το υποστηριζόμενο εύρος είναι από '1000-01-01' μέχρι '9999-12-31'
DATETIME()	Μορφή : YYYY-MM-DD HH:MM:SS **Note: Προσοχή:** Το υποστηριζόμενο εύρος είναι από '1000-01-01 00:00:00' μέχρι '9999-12-31 23:59:59'
TIMESTAMP()	Οι TIMESTAMP τιμές αποθηκεύονται ως ο αριθμός των δευτερολέπτων σύμφωνα με το σύστημα UNIX ('1970-01-01 00:00:00' UTC). Μορφή : YYYY-MM-DD HH:MM:SS **Προσοχή :** Το υποστηριζόμενο εύρος είναι από '1970-01-01 00:00:01' UTC μέχρι '2038-01-09 03:14:07' UTC
TIME()	Μορφή : HH:MM:SS **Προσοχή :** Το υποστηριζόμενο εύρος είναι από '-838:59:59' μέχρι '838:59:59'
YEAR()	Έτος σε μορφή δυο ή τεσσάρων αριθμών. **Προσοχή :** Το υποστηριζόμενο εύρος είναι για μορφή τεσσάρων αριθμών: 1901 μέχρι 2155. Το υποστηριζόμενο εύρος για μορφή δυο αριθμών:: 70 μέχρι 69, αναπαριστώντας τα έτη 1970 μέχρι 2069

* Αν και η DATETIME και η TIMESTAMP επιστρέφουν την ίδια μορφή, λειτουργούν πολύ διαφορετικά. Σε ένα ερώτημα INSERT ή UPDATE, η TIMESTAMP παίρνει αυτόματα την τρέχουσα ημερομηνία και ώρα. Η TIMESTAMP δέχεται επίσης διάφορες μορφές, όπως YYYYMMDDHHMMSS , YYMMDDHHMMSS , YYYYMMDD, ή YYMMDD.

SQL

ADVANCED

ΧΡΗΣΤΕΣ

Οι βάσεις δεδομένων περιέχουν ένα σύνολο από χρήστες. Οι χρήστες αυτοί είναι διαφορετικοί από τους χρήστες που διαχειρίζεται το λειτουργικό σύστημα στο οποίο τρέχει ο server. Οι χρήστες κατέχουν αντικείμενα της βάσης δεδομένων (για παράδειγμα, πίνακες) και μπορούν να εκχωρήσουν προνόμια σε άλλους χρήστες για αυτά τα αντικείμενα έτσι ώστε να ελέγχουν ποιος έχει πρόσβαση σε ποιο αντικείμενο.

Δημιουργία χρήστη

Σύνταξη :

```
CREATE USER ‘users_name’@’host’
```

Παράδειγμα :

```
CREATE USER ‘mits’@’localhost’
```

Δημιουργία χρήστη με κωδικό

Σύνταξη :

```
CREATE USER users_name IDENTIFIED BY 'password';
```

Παράδειγμα :

```
CREATE USER ‘mits’@’localhost’
IDENTIFIED BY ‘12Ab$%';
```

Διαγραφή χρήστη

Σύνταξη :

```
DROP USER ‘users_name’@’host’
```

Παράδειγμα :

```
DROP USER ‘mits’@’localhost’
```

* Εάν καθορίσετε μόνο το όνομα χρήστη πχ CREATE USER mits, τότε χρησιμοποιείται το όνομα «%» για τον host . Δηλαδή CREATE USER ‘users_name’@’%’.

ΤΕΛΟΣ!!!