Οδηγός για Δημιουργία και Διαχείριση Βάσης Δεδομένων σε MySQL και SQL Server

Αυτό το έγγραφο αποτελεί μια συνοπτική αλλά πλήρη παρουσίαση των βασικών εντολών SQL που χρειάζονται οι φοιτητές για να δημιουργήσουν και να διαχειριστούν μία βάση δεδομένων. Περιλαμβάνει τόσο θεωρία όσο και πρακτική χρήση, καθώς και όλες τις βασικές διαφορές μεταξύ MySQL και SQL Server.

---

1. Η εντολή CREATE TABLE

Η εντολή CREATE TABLE χρησιμοποιείται για τη δημιουργία νέου πίνακα. Η γενική της δομή (και στα δύο συστήματα) είναι:

```
CREATE TABLE table_name (
  column_name1 data_type [constraints],
  column_name2 data_type [constraints],
  ...
  table_constraints
);
```

Γενικοί κανόνες:

- Κάθε πίνακας πρέπει να έχει **PRIMARY KEY**.
- Χρησιμοποιούμε **κατάλληλους τύπους δεδομένων** (INT, VARCHAR, DATE κτλ.).
- Οι περιορισμοί (constraints) μπορούν να μπουν σε κάθε στήλη ή στο τέλος του πίνακα.
- Το όνομα του πίνακα και των στηλών **δεν πρέπει να έχει κενά**.
- Προτιμάμε **μικρά γράμματα με underscores** (π.χ. student_id).

MySQL

```
CREATE TABLE person (
  personID INT NOT NULL AUTO_INCREMENT,
  lastName VARCHAR(30) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
```

```
  birthdate DATE NOT NULL,
  address VARCHAR(50),
  city VARCHAR(30),
  PRIMARY KEY (personID)
);
```

SQL Server

```
CREATE TABLE person (
  personID INT IDENTITY(1,1) NOT NULL,
  lastName VARCHAR(30) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
  birthdate DATE NOT NULL,
  address VARCHAR(50),
  city VARCHAR(30),
  PRIMARY KEY (personID)
);
```

Κύρια διαφορά:

- MySQL: AUTO_INCREMENT
- SQL Server: IDENTITY(1,1)

---

2. Περιορισμοί (Constraints): Τι είναι και πώς χρησιμοποιούνται

Οι περιορισμοί χρησιμοποιούνται ώστε τα δεδομένα στους πίνακες να είναι σωστά και έγκυρα.

Κάθε constraint μπορεί να χρησιμοποιηθεί:

- **σε επίπεδο στήλης** (δίπλα από τον τύπο δεδομένων)
- **σε επίπεδο πίνακα** (στο τέλος του CREATE TABLE)

Παρακάτω δίνεται τι κάνει ο κάθε περιορισμός και πώς χρησιμοποιείται.

2.1 NOT NULL

Τι κάνει:

Απαιτεί η στήλη να έχει πάντα τιμή.

Πώς χρησιμοποιείται:

```
name VARCHAR(50) NOT NULL
```

2.2 UNIQUE

Τι κάνει:

Οι τιμές της στήλης πρέπει να είναι διαφορετικές για κάθε εγγραφή.

Πώς χρησιμοποιείται:

```
email VARCHAR(50) UNIQUE
```

Ή:

```
CONSTRAINT UQ_email UNIQUE (email)
```

2.3 PRIMARY KEY

Τι κάνει:

Ορίζει τη μοναδική ταυτότητα κάθε εγγραφής.

Πώς χρησιμοποιείται:

```
PRIMARY KEY (personID)
```

2.4 FOREIGN KEY

Τι κάνει:

Δηλώνει ότι η τιμή της στήλης αντιστοιχεί σε τιμή άλλου πίνακα.

Πώς χρησιμοποιείται:

MySQL:

FOREIGN KEY (personID) REFERENCES person(personID)

SQL Server:

CONSTRAINT FK_orders_person FOREIGN KEY (personID)

REFERENCES person(personID)

2.5 CHECK

Τι κάνει:

Επιβάλλει συνθήκη που πρέπει να ισχύει για τη στήλη.

Πώς χρησιμοποιείται:

```
age INT CHECK (age >= 18)
```

2.6 DEFAULT

Τι κάνει:

Ορίζει προεπιλεγμένη τιμή.

Πώς χρησιμοποιείται:

```
country VARCHAR(30) DEFAULT 'Greece'
```

---

3. FOREIGN KEY — Όλες οι διαθέσιμες επιλογές

Οι επιλογές αυτές δηλώνουν τι θα γίνει στις εγγραφές του παιδικού πίνακα όταν διαγραφεί/αλλάξει μια εγγραφή στον γονικό πίνακα.

CASCADE

Διαγράφει ή ενημερώνει αυτόματα τις συνδεδεμένες εγγραφές.

SET NULL

Ορίζει NULL το πεδίο του ξένου κλειδιού.

SET DEFAULT

Ορίζει την default τιμή της στήλης.

RESTRICT (MySQL)

Απαγορεύει διαγραφή/αλλαγή.

NO ACTION (default)

Όμοιο με RESTRICT.

Πλήρες παράδειγμα MySQL:

```
FOREIGN KEY (personID) REFERENCES person(personID)
  ON DELETE CASCADE
  ON UPDATE SET NULL;
```

Πλήρες παράδειγμα SQL Server:

```
CONSTRAINT FK_orders_person FOREIGN KEY (personID)
  REFERENCES person(personID)
  ON DELETE CASCADE
  ON UPDATE CASCADE;
```

---

4. Βασικές εντολές δεδομένων (DML)

INSERT

Η εντολή εισάγει νέα εγγραφή.

```
INSERT INTO person (lastName, firstName, birthdate, address, city)
VALUES ('John', 'Doe', '2000-01-31', 'Pavlou Mela', 'Kozani');
```

UPDATE

Αλλάζει τιμές υπαρχουσών εγγραφών.

```
UPDATE person
SET city = 'Kozani'
```

```
WHERE personID = 5;
```

DELETE

Διαγράφει εγγραφές.

```
DELETE FROM person WHERE personID = 5;
```

SELECT

Ανακτά δεδομένα.

```
SELECT * FROM person;
```

---

5. ALTER TABLE — Όλες οι επιλογές και πώς χρησιμοποιούνται

Η εντολή ALTER TABLE αλλάζει τη δομή ενός πίνακα.

5.1 Προσθήκη νέας στήλης

```
ALTER TABLE person ADD phone VARCHAR(20);
```

5.2 Διαγραφή στήλης

```
ALTER TABLE person DROP COLUMN phone;
```

5.3 Αλλαγή τύπου ή μεγέθους στήλης

MySQL:

```
ALTER TABLE person MODIFY COLUMN city VARCHAR(40);
```

SQL Server:

```
ALTER TABLE person ALTER COLUMN city VARCHAR(40);
```

5.4 Μετονομασία στήλης

MySQL:

```
ALTER TABLE person RENAME COLUMN oldName TO newName;
```

SQL Server:

```
EXEC sp_rename 'person.oldName', 'newName', 'COLUMN';
```

5.5 Προσθήκη περιορισμού

```
ALTER TABLE person
ADD CONSTRAINT CHK_age CHECK (age >= 18);
```

5.6 Διαγραφή περιορισμού

MySQL:

```
ALTER TABLE person DROP CHECK CHK_age;
```

SQL Server:

```
ALTER TABLE person DROP CONSTRAINT CHK_age;
```

5.7 Προσθήκη FOREIGN KEY

```
ALTER TABLE orders
ADD CONSTRAINT FK_orders_person
FOREIGN KEY (personID) REFERENCES person(personID)
ON DELETE CASCADE;
```

5.8 Διαγραφή FOREIGN KEY

MySQL:

```
ALTER TABLE orders DROP FOREIGN KEY FK_orders_person;
```

SQL Server:

```
ALTER TABLE orders DROP CONSTRAINT FK_orders_person;
```

---

6. Διαγραφή πίνακα

Κοινό και στα δύο συστήματα:

```
DROP TABLE person;
```