Άλλες εντολές SQL

Εντολές τροποποίησης δεδομένων

- Η SELECT δεν επηρεάζει τα αποθηκευμένα δεδομένα
- Εισαγωγή, διαγραφή, τροποποίηση: **αλλάζουν** τα δεδομένα στη βάση
- Υπάρχει δυνατότητα για αναίρεση πριν το κλείσιμο και την οριστική αποθήκευση (rollback)

Εισαγωγή νέων δεδομένων

- Εισαγωγή σημαίνει **ΝΕΕΣ ΓΡΑΜΜΕΣ**
- Εφαρμόζονται όποιοι έλεγχοι απαιτείται
  - Μοναδικές τιμές κλειδιού
  - Τύποι δεδομένων
  - Κενές τιμές

INSERT INTO <πίνακας> [(<λίστα πεδίων>)]

VALUES (< λίστα τιμών>)

Εντολή INSERT

- Παράδειγμα: εισαγωγή γραμμής στον πίνακα Borrow με αριθμό δανείου και όνομα πελάτη, με κενά τα υπόλοιπα πεδία

  INSERT INTO Borrow (cust-name, loan-no)

  VALUES (‘Πέτρου’, 23910)

- Οι τιμές αντιστοιχούν μία προς μία στα πεδία όπως αυτά γράφονται στην εντολή
- Πρέπει να περιλαμβάνονται τουλάχιστον οι μη κενές (NOT NULL) τιμές, αλλιώς η εντολή δεν δουλεύει

Εντολή INSERT

- Παράδειγμα: εισαγωγή γραμμής στον πίνακα Borrow με αριθμό δανείου και όνομα πελάτη, με κενά τα υπόλοιπα πεδία

  INSERT INTO Borrow (cust-name, loan-no)

  VALUES (‘Πέτρου’, 23910)

- Οι τιμές αντιστοιχούν μία προς μία στα πεδία όπως αυτά γράφονται στην εντολή
- Πρέπει να περιλαμβάνονται τουλάχιστον οι μη κενές (NOT NULL) τιμές, αλλιώς η εντολή δεν δουλεύει

Εντολή INSERT

- Μπορεί να παραλειφθεί η λίστα πεδίων, αρκεί να δίνονται όλες οι τιμές της γραμμής, **με τη σειρά που έχουν ορισθεί στον πίνακα**

```
INSERT INTO Branch
VALUES (‘Κέντρο’, ‘Αθήνα’, 1000000)
```

- Μπορεί να ζητήσουμε να μείνει ένα πεδίο κενό

```
INSERT INTO Borrow
VALUES (12555, null, null, 1000.00)
```

Εντολή INSERT

- Υπάρχει και η δυνατότητα να δώσουμε «πακέτο» το αποτέλεσμα μιας εντολή SELECT (προσοχή: συμβατότητα)
- Παράδειγμα: δημιουργούμε έναν νέο πίνακα Branch_Athens, με στήλες br-name, assets και θέλουμε να εισάγουμε αυτά τα στοιχεία για τα καταστήματα της Αθήνας
- Καταστήματα Αθήνας: SELECT br-name, assets FROM branch

  WHERE br-city = ‘Αθήνα’
- Εισαγωγή στον νέο πίνακα

  INSERT INTO Branch_Athens

  (SELECT br-name, assets FROM branch WHERE br-city = ‘Αθήνα’)

Εντολή INSERT: παραδείγματα

- Εισαγωγή του προμηθευτή S5, με στάτους 3 και πόλη Αθήνα

  INSERT INTO S (S#, status, city)

  VALUES (‘S5’, 3 , ‘Αθήνα’)
- Εναλλακτικά INSERT INTO S VALUES (‘S5’, null, ‘Αθήνα’, 3) (ακολουθούμε τη σειρά των πεδίων στον πίνακα)
- Προσοχή: όχι εισαγωγικά στο null, γιατί θα θεωρηθεί λέξη
  - Παρά ταύτα, η εντολή θα τρέξει, βάζοντας σαν Sname το null

- Παραγγελία από S1, για το υλικό P2, στο έργο J5 , χωρίς να ορίσουμε ποσότητα

  INSERT INTO SPJ VALUES (‘S1’, ‘P2’, ‘J5’, null)

  ή INSERT INTO SPJ (S#, P#, J#) VALUES (‘S1’, ‘P2’, ‘J5’)

Εντολή INSERT: παραδείγματα

- Εισαγωγή στις παραγγελίες όλων των πιθανών συνδυασμών S#, P#, J# χωρίς τιμή στην ποσότητα
- INSERT INTO SPJ (S#, P#, J#)

  SELECT S#, P#, J# FROM S, P, J

Χωρίς συνθήκη σύνδεσης, **καρτεσιανό γινόμενο**!

- Εναλλακτικά

  INSERT INTO SPJ

  SELECT S#, P#, J#, null FROM S, P, J

- Θυμηθείτε: στη SELECT μπορούμε να έχουμε και εκφράσεις ή και απλές τιμές (το null είναι η απουσία τιμής, αλλά εδώ α΄πλώς σχηματίζει την τετράδα για τη γραμμή του SPJ)

Διαγραφή δεδομένων

- Διαγραφή: Σβήσιμο **ολόκληρων γραμμών** (όχι τιμών)
- Πρώτα αναζήτηση – μετά σβήσιμο

  DELETE <πίνακας>

  [WHERE <συνθήκη>]
- ΠΡΟΣΟΧΗ: δεν περιλαμβάνεται στο standard
- Μπορεί να τη δούμε και με τη μορφή DELETE **FROM**

- Παράδειγμα: διαγραφή όλων των δανείων του πελάτη Πέτρου

  DELETE Borrow

  WHERE Cust-name = ‘Πέτρου’

Εντολή DELETE

- Η συνθήκη μπορεί να είναι ο,τιδήποτε
- Το WHERE λειτουργεί ακριβώς όπως και στη SELECT (θυμηθείτε: τουβλάκια)

```
DELETE Customer
WHERE Cust-name LIKE ‘A%’ (τι κάνει;)
```

- Απάντηση: Διαγράφει πελάτες με όνομα που αρχίζει από Α

```
DELETE Borrow
WHERE amount < 1000 AND amount >500
```

Εντολή DELETE

- Το WHERE δεν είναι απαραίτητο για να λειτουργήσει η εντολή:

  DELETE Borrow

  (τι πιστεύετε ότι κάνει; Σκεφτείτε)

- Απάντηση: διαγραφή **όλων των γραμμών** (όχι του πίνακα, **η δομή παραμένει**)

Ενημέρωση δεδομένων

- Ενημέρωση: αλλαγή δεδομένων που υπάρχουν ήδη
- Όχι προσθήκη/διαγραφή γραμμών

  UPDATE <πίνακας>

  SET <λίστα αναθέσεων>

  [WHERE <συνθήκη>]

Εντολή UPDATE: τεχνικές λεπτομέρειες

- Μπορώ να κάνω περισσότερες αλλαγές ταυτόχρονα (πάντα στην ίδια γραμμή/ές)
- Μπορώ να κάνω υπολογισμό και με το αποτέλεσμα να κάνω ενημέρωση

```
UPDATE Borrow
SET Br-name = ‘Ομόνοια’ , amount = amount * 1,2
WHERE Br-name = ‘Κέντρο’
```

Ενημέρωση δεδομένων: UPDATE

- Αλλαγή του ποσού του δανείου του πελάτη Πέτρου

  UPDATE Borrow

  SET amount = 2000

  WHERE Cust-name = ‘Πέτρου’
- Αλλαγή του ονόματος ενός πελάτη με ένα νέο

  UPDATE Borrow

  SET Cust-name = ‘Πετρίδης’

  WHERE Cust-name = ‘Πέτρου’

Ενημέρωση δεδομένων: UPDATE

- Προσοχή στη συνθήκη: αν δεν μπει,…

  UPDATE Borrow

  SET amount = 2000

- Απάντηση: η αλλαγή εφαρμόζεται **σε όλες τις γραμμές του πίνακα!**

Εντολές ορισμού δεδομένων

- Ενέργειες για τις δομές που δημιουργούμε σε μια βάση δεδομένων
  - Δημιουργία πίνακα
  - Τροποποίηση πίνακα
  - Διαγραφή πίνακα
- Δεν ασχολούνται με δεδομένα
- Δημιουργούν/τροποποιούν δομές

Δημιουργία πίνακα

CREATE TABLE <όνομα πίνακα>
(<όνομα πεδίου> <τύπος πεδίου> [NOT NULL],
<όνομα πεδίου> <τύπος πεδίου> [NOT NULL],
…
<όνομα πεδίου> <τύπος πεδίου> [NOT NULL],
[PRIMARY KEY (<λίστα πεδίων>),]
[FOREIGN KEY (<όνομα πεδίου>) REFERENCES <όνομα πίνακα>])

Δημιουργία πίνακα

- Πρέπει να ορίσουμε τύπο για κάθε χαρακτηριστικό
- Τύποι: Real, float, number, integer (int), date, varchar (n), logical, Boolean, …
- NOT NULL: δεν είναι υποχρεωτικό, χρειάζεται για να είμαστε βέβαιοι ότι θα δίνεται τιμή στο χαρακτηριστικό αυτό σε όλες τις γραμμές
- NULL: κενή τιμή, όχι 0 ή spacebar

Δημιουργία πίνακα

- Πρωτεύον κλειδί (primary key): μοναδικές τιμές, ποτέ κενό
- Μπορεί να είναι και πάνω από ένα πεδίο

- Ξένο κλειδί (foreign key): οι τιμές σε αυτό το χαρακτηριστικό πρέπει να υπάρχουν ήδη στο χαρακτηριστικό που αναφέρεται στο REFERENCES

Δημιουργία πίνακα

- Για τον πίνακα SPJ:

```
CREATE TABLE SPJ
(S# Varchar (6) NOT NULL,
  P# Varchar (6) NOT NULL,
  J# Varchar (6) NOT NULL,
  posot integer,
PRIMARY KEY (S#,P#,J#),
 FOREIGN KEY (S#) REFERENCES S (S#),
 FOREIGN KEY (J#) REFERENCES J (J#),
 FOREIGN KEY (P#) REFERENCES P (P#)  )
```

Δημιουργία πίνακα: παρατηρήσεις (1)

- Ο τύπος δεδομένων σχεδόν απαραίτητος: αν δεν ορισθεί, το σύστημα βάζει κάποιον default, πχ char(1), που είναι άχρηστος
- Not null: αυτό το πεδίο πρέπει να έχει πάντα τιμή
  - Το κρίνει ο σχεδιαστής
  - Πχ. Στη Γραμματεία θέλουμε σίγουρα ΑΜ και όνομα, αλλά ξέρουμε ότι πάντα ο φοιτητής βρίσκεται σε κάποιο εξάμηνο, άρα θέλουμε και το εξάμηνο not null
  - Γενικά πάντως δεν βάζουμε not null τυχαία, πρέπει να στηριζόμαστε κάπου

Δημιουργία πίνακα: παρατηρήσεις (2)

- Πρωτεύον κλειδί: μοναδικές τιμές, όχι κενά
  - Αν χρειάζεται, ζευγάρι (ή τριάδα) τιμών: **ένα** κλειδί, με **δύο μέρη**
- Ξένο κλειδί: πεδίο που το «παίρνουμε» από κάποιον άλλο πίνακα
  - S# στον SPJ: για να έχει παραγγελίες, υπάρχει σίγουρα στον πίνακα S
  - Αν η τιμή που θέλουμε να βάλουμε σε τέτοιο πεδίο δεν υπάρχει στο αρχικό πίνακα, το σύστημα δεν μας το επιτρέπει
  - Το REFERENCES δείχνει σε ποιον πίνακα κοιτάμε και σε ποιο πεδίο εκείνου του πίνακα

Διαγραφή πίνακα

- Διαγραφή όλου του πίνακα, τόσο δεδομένα όσο και δομή

DROP TABLE <όνομα πίνακα>

- Διαγραφή όλου του πίνακα, τόσο δεδομένα όσο και δομή
- Ιδιαίτερη προσοχή – πάντα το σύστημα προειδοποιεί