ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΜΕΡΟΣ ΤΕΤΑΡΤΟ

Insert, Update, Delete, Ένωση πινάκων

Γιώργος Μαρκομανώλης

Περιεχόμενα

Group By 1
Having 1
Order By 2
Εντολή Insert 3
Εντολή Update 4
Εντολή Delete 5
Κατηγόρημα Exists 5
Υποερωτήματα 5
Inner Join 6
Outer Join 7
Καρτεσιανό γινόμενο 8
Ψευδώνυμα πινάκων 8
Υποερωτήματα για σύνδεση πινάκων 8
Union 9

Παράμετροι μετά το Where

Μια απλή γενική σύνταξη της **SELECT** για ανάκτηση πληροφοριών από έναν πίνακα, είναι η ακόλουθη:

```
SELECT ονόματα στηλών ή *
FROM όνομα πίνακα
WHERE συνθήκες
παράμετροι;
```

GROUP BY

Με την παράμετρο **GROUP BY** μπορείτε να χωρίσετε τα αποτελέσματα που θα εμφανιστούν ανά ομάδες.

Παράδειγμα

Αν έχετε τον παρακάτω πίνακα με τα ακόλουθα δεδομένα,

Τύπος Προϊόν (E_TYPE)	Τιμή (PRICE)
Παπούτσια	40
Παπούτσια	50
Παλτό	150

```
SELECT  E_TYPE,SUM(PRICE)
FROM PRODUCTS
GROUP BY E_TYPE;
```

Με την παραπάνω εντολή θα εμφανιστεί ο παρακάτω πίνακας όπου έχει γίνει ομαδοποίηση με βάση τον τύπο προϊόν που βάλαμε μετά το **GROUP BY** και στην στήλη των τιμών έχει το άθροισμα των τιμών ανά είδος.

Τύπος Προϊόν (E_TYPE)	Τιμή (PRICE)
Παπούτσια	90
Παλτό	150

Αντί για SUM μπορείτε να χρησιμοποιήσετε και άλλες συναρτήσεις.

Προσοχή: Οι στήλες που επιλέγονται πρέπει να μπορούν να ομαδοποιηθούν, δηλαδή να υπάρχουν ίδιες τιμές για κάποιο γνώρισμα.

HAVING

Η παράμετρος **HAVING** χρησιμοποιείται μετά το **GROUP BY** και προσθέτει επιπλέον συνθήκη για την εύρεση δεδομένων. Δηλαδή αν θέλαμε μια ομαδοποίηση δεδομένων

όπως κάναμε πριν αλλά να εμφανιστούν μόνο όσες πωλήσεις είναι πάνω από 100, τότε πρέπει να δώσουμε την εντολή:

```
SELECT  E_TYPE,SUM(PRICE)
FROM PRODUCTS
GROUP BY E_TYPE
HAVING SUM(PRICE)>100;
```

Με την παραπάνω εντολή θα εμφανιστεί ο πίνακας με τις πωλήσεις άνω του 100.

Τύπος Προϊόν (E_TYPE)	Τιμή (PRICE)
Παλτό	150

Προσοχή: Μετά την παράμετρο **HAVING** μπορείτε να γράψετε στήλες ή συναρτήσεις στηλών που έχουν ομαδοποιηθεί.

ORDER BY

Η παράμετρος **ORDER BY** χρησιμοποιείται για την ταξινόμηση των εγγραφών κατά την ανάκτηση από την βάση.

Αν θέλαμε να δούμε τα δεδομένα του πίνακα ταξινομημένα με βάση τον τύπο του προϊόν, θα δίναμε την εντολή:

```
SELECT *
FROM PRODUCTS
ORDER BY E_TYPE;
```

Εντολή Insert

Με την εντολή **Insert** καταχωρείτε δεδομένα στους πίνακες μιας βάσης. Όπως έχουμε ήδη αναφέρει και σε προηγούμενο μάθημα η σύνταξη της εντολής είναι η εξής:

INSERT INTO <όνομα πίνακα ή όνομα όψης> (ονόματα στηλών για τα οποία θα εισάγετε δεδομένα)
VALUES (δεδομένα που θα εισάγετε για κάθε στήλη χωρισμένα με κόμμα)

Παράδειγμα

Για να καταχωρήσουμε ένα νέο τύπο προϊόν με όνομα Καπέλα και με πωλήσεις 30 στον πίνακα **PRODUCTS**, δίνουμε την εντολή:

INSERT INTO PRODUCTS (E_TYPE,PRICE) VALUES ('Καπέλα','30');

Αν έχουμε δύο πίνακες και θέλουμε να καταχωρήσουμε στον δεύτερο τα περιεχόμενα του πρώτου, δίνουμε την εντολή:

INSERT INTO ΠΙΝΑΚΑ_Β(Στήλη1, Στήλη2, ...)
SELECT Στήλη1, Στήλη2, Στήλη3 FROM ΠΙΝΑΚΑ_Α;

Μετά την εντολή **Insert** δίνουμε την εντολή

COMMIT;

Για να καταχωρηθούν τα δεδομένα στην βάση.

Παρατήρηση: Όταν έχετε ορίσει σε μια στήλη Default τιμή, τότε για να καταχωρηθεί η Default τιμή πρέπει να μην βάλετε ούτε το όνομα της στήλης αλά ούτε και κάποια τιμή για αυτή την στήλη στην εντολή Insert.

Εντολή Update

Με την εντολή **Update** μπορούμε να αλλάξουμε τα δεδομένα που ήδη έχουν καταχωρηθεί στην βάση. Η σύνταξη της εντολής είναι η ακόλουθη:

UPDATE όνομα_πίνακα | όνομα_όψης
SET όνομα_στήλης = τιμή
[WHERE συνθήκες αναζήτησης]

Η νέα τιμή της στήλης μπορεί να είναι:

- NULL, αν γίνεται η στήλη να είναι NULL.
- Μια σταθερή τιμή
- Αν θέλουνε να καταχωρήσουμε κείμενο 'Γιάννης' που έχει διαφορετικό Character Set από αυτό που έχει δημιουργηθεί η στήλη, μπορούμε να δώσουμε την εντολή:

  SET όνομα_στήλης = _IS08859_7 'Γιάννης'
- Μπορούμε να ορίσουμε μια στήλη να έχει τις τιμές που έχει μια άλλη στήλη με την εντολή:

  SET όνομα_στήλης_A = όνομα_στήλης_B
- Αν έχουμε μια στήλη με τύπο δεδομένων ημερομηνία και θέλουμε να την αλλάξουμε και να βάλουμε μια ημερομηνία, δύο βδομάδες αργότερα, δίνουμε την εντολή:

  SET E_DATE=E_DATE+14
- Φυσικά μπορούμε να χρησιμοποιήσουμε εντολές από τον server, δηλαδή αν θέλουμε να καταχωρήσουμε την σημερινή ημερομηνία, δίνουμε την εντολή:

  SET E_DATE=CURRENT_DATE

Μπορούμε να κάνουμε πράξεις με τις τιμές στηλών, δηλαδή αν θέλουμε η στήλη_Α να έχει τιμή την στήλη_Α και την στήλη_Β, τότε δίνουμε την εντολή:

SET Στήλη_A=Στήλη_A+Στήλη_B

Εντολή Delete

Η εντολή **Delete** χρησιμεύει για την διαγραφή γραμμών από έναν πίνακα. Η σύνταξη της εντολής είναι η ακόλουθη:

DELETE FROM όνομα_πίνακα
WHERE συνθήκες αναζήτησης;

Αν δεν μπουν συνθήκες αναζήτησης, τότε θα σβηστούν όλες οι γραμμές του πίνακα.

Παράδειγμα

Για να διαγράψουμε τον προϊόν 'Καπέλα', δίνουμε την εντολή:

DELETE FROM PRODUCTS
WHERE E_TYPE='Καπέλα';

Κατηγόρημα Exists

Με το κατηγόρημα **Exists** μπορούμε να ελέγξουμε αν ισχύει ένα ερώτημα.

Παράδειγμα

Θα κάνουμε Update τον πίνακα PRODUCTS στο προϊόν που έχει όνομα Καπέλα, αρκεί να υπάρχει γραμμή που να έχει τιμή 30 και θα την κάνουμε 40. Δίνουμε την εντολή:

```
UPDATE PRODUCTS
SET PRICE=40
WHERE E_TYPE='Καπέλα'
AND EXISTS(SELECT SALES FROM PRODUCTS WHERE PRICE=30);
```

Υποερωτήματα

Μπορείτε μέσα σε μια εντολή **SELECT** να έχετε και άλλη εντολή **SELECT**. Δηλαδή αν θέλουμε να βρούμε το προϊόν που έχει την μέγιστη τιμή πώλησης, τότε δίνουμε την εντολή:

```
 SELECT E_TYPE
FROM PRODUCTS
WHERE PRICE=(SELECT MAX(PRICE) FROM PRODUCTS);
```

Ερωτήσεις σε πολλούς πίνακες

Η Firebird μπορεί να κάνει ερωτήσεις σε πολλούς πίνακες με τις μεθόδους: joins, subqueries και unions. Αυτά τα ερωτήματα έχουν ως σκοπό την εύρεση δεδομένων που δεν βρίσκονται μόνο σε έναν πίνακα αλλά σε δύο ή και περισσότερους πίνακες.

Inner Join

Όταν σε μια εντολή **SELECT** θέλουμε να βρούμε δεδομένα από δύο πίνακες (ή και περισσότερους) τότε για να ορίσετε μια στήλη στην εντολή, γράφετε πρώτα το όνομα του πίνακα, μετά τελεία και μετά το όνομα της στήλης χωρίς κενά.

Έστω ότι έχουμε δύο πίνακες με τα παρακάτω δεδομένα:

PRODUCTS

E_TYPE	PRICE
Παλτό	150
Παπούτσια	50

STOCK

E_TYPE	E_DATE
Παλτό	5/11/204
Παπούτσια	11/11/2004

Στον πίνακα Products έχουμε τα προϊόντα και την τιμή πώλησης και στον πίνακα STOCK έχουμε τα προϊόντα και την ημερομηνία παραλαβής τους. Στον πίνακα Products η στήλη E_TYPE είναι primary key και στον πίνακα STOCK η στήλη E_TYPE είναι foreign key στην στήλη E_TYPE του πίνακα Products.

Αν θέλουμε να βρούμε τα προϊόντα που έχουμε, την τιμή τους και την ημερομηνία παραλαβής τότε δίνουμε την εντολή:

```
SELECT PRODUCTS.E_TYPE, PRICE, E_DATE
FROM PRODUCTS INNER JOIN STOCK
ON PRODUCTS.E_TYPE=STOCK.E_TYPE;
```

Θα εμφανιστεί ο ακόλουθος πίνακας:

E_TYPE	PRICE	E_DATE
Παλτό	150	5/11/2004
Παπούτσια	50	11/11/2004

Παρατήρηση: Αυτή η σύνδεση πινάκων λειτουργεί κανονικά, όταν ο πίνακας πριν την εντολή **INNER JOIN** έχει primary key και ο πίνακας μετά από αυτή την εντολή έχει foreign key που αναφέρεται στο primary key του πρώτου πίνακα.

Επίσης αν στον πίνακα Products είχαμε ένα επιπλέον προϊόν που δεν υπήρχε στον πίνακα STOCK, τότε δεν θα εμφανιζόταν καθόλου αυτό το προϊόν, δηλαδή πρέπει να υπάρχουν τιμές και στους δύο πίνακες.
Μπορούμε φυσικά να βάλουμε και συνθήκη **Where** μετά την συνθήκη **ON**.
Η λέξη **INNER** στην συνθήκη **INNER JOIN** είναι προαιρετική.

Αν θέλετε να συνδέσετε 3 πίνακες, τότε μπορούμε να δώσουμε ένα **SELECT** της ακόλουθης μορφής:

```
SELECT Table1.ColumnA, Table2.ColumnB, Table3.ColumnC
FROM Table1 JOIN Table2
ON Table1.ColumnB=Table2.ColumnD
JOIN Table3 ON Table2.ColumnE=Table3.ColumnD
WHERE συνθήκες αναζήτησης;
```

Outer Join

Σε αντίθεση με το INNER JOIN, η σύνδεση OUTER JOIN θα εμφανίσει όλα τις εγγραφές του πίνακα ανεξάρτητα αν υπάρχουν τιμές στον άλλον πίνακα βάζοντας NULL όπου δεν υπάρχουν τιμές.

Δηλαδή αν προσθέσουμε στον πίνακα products το προϊόν Καπέλα με τιμή 20, τότε αν δώσουμε την εντολή:

```
SELECT PRODUCTS.E_TYPE, PRICE, E_DATE
FROM PRODUCTS LEFT OUTER JOIN STOCK
ON PRODUCTS.E_TYPE=STOCK.E_TYPE;
```

Θα εμφανιστεί ο πίνακας:

E_TYPE	PRICE	E_DATE
Καπέλα	20	NULL
Παλτό	150	5/11/2004
Παπούτσια	40	11/11/2004

Συμβουλή: Για να θυμάστε το **LEFT OUTER JOIN** να σκέφτεστε ότι με το left δηλώνουμε ότι θα εμφανιστούν όλες οι εγγραφές του αριστερού πίνακα.
Αντίστοιχα για **RIGHT OUTER JOIN** θα εμφανιστούν όλες οι τιμές του δεξιού πίνακα.
Για το **FULL JOIN** θα εμφανιστούν όλες οι τιμές και από τους δύο πίνακες συνδυάζοντας το LEFT και το RIGHT OUTER JOIN.

Καρτεσιανό γινόμενο

Για να ανακτήσουμε το καρτεσιανό γινόμενο των παραπάνω πινάκων, δίνουμε την εντολή:

SELECT PRODUCTS.*, STOCK.* FROM PRODUCTS, STOCK;

Θέλουμε όμως να πάρουμε την ένωση των πινάκων, άρα δίνουμε την εντολή:

SELECT PRODUCTS.*, STOCK.E_DATE
FROM PRODUCTS, STOCK WHERE
PRODUCTS.E_TYPE=STOCK.E_TYPE;

Με την παραπάνω εντολή θα πάρουμε τον πίνακα:

E_TYPE	PRICE	E_DATE
Παλτό	150	5/11/2004
Παπούτσια	40	11/11/2004

Ψευδώνυμα πινάκων

Αν τα ονόματα των πινάκων είναι μεγάλα και θέλετε να χρησιμοποιείτε άλλα ονόματα για να δίνετε εντολές, πρέπει να δηλώσουμε ψευδώνυμα στους πίνακες.
Για την παραπάνω εντολή θα μπορούσαμε να την γράψουμε ως εξής:

SELECT A.*, B.E_DATE
FROM PRODUCTS A, STOCK B
A.E_TYPE=B.E_TYPE

Υποερωτήματα για σύνδεση πινάκων

Έχουμε υποερωτήματα αν δίνουμε εντολή **SELECT** που περιέχει και άλλη εντολή **SELECT**. Δηλαδή με την εντολή:

SELECT PRODUCTS.*,(SELECT E_DATE FROM STOCK WHERE
PRODUCTS.E_TYPE=STOCK.E_TYPE)
FROM PRODUCTS, STOCK
WHERE PRODUCTS.E_TYPE=STOCK.E_TYPE;

Με την παραπάνω εντολή θα πάρουμε τον πίνακα:

E_TYPE	PRICE	E_DATE
Παλτό	150	5/11/2004
Παπούτσια	40	11/11/2004

Union

Με την εντολή **UNION** μπορούμε να ενώσουμε δύο ή περισσότερες εντολές **SELECT.** Δηλαδή αν δώσουμε την εντολή:

SELECT * FROM PRODUCTS WHERE PRICE='50' UNION SELECT * FROM PRODUCTS WHERE E_TYPE='Καπέλα';

Θα εμφανιστεί ο πίνακας που είναι η ένωση των δύο **SELECT**, δηλαδή:

E_TYPE	PRICE
Καπέλα	20

Ενώ αν δώσουμε την εντολή:

SELECT * FROM PRODUCTS WHERE PRICE='150' UNION SELECT * FROM PRODUCTS WHERE E_TYPE='Καπέλα';

Θα εμφανιστεί ο πίνακας:

E_TYPE	PRICE	E_DATE
Καπέλα	20	NULL
Παλτό	150	5/11/2004