ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

Η Γλώσσα SQL

Χρυσόστομος Καπέτης

Επισκόπηση

- Η γλώσσα SQL
- Εντολές DML (Data Manipulation Language)
- Εντολές DDL (Data Definition Language)
- Συναθροιστικές συναρτήσεις (Aggregate functions)
- Κενές τιμές (Null Values)
- Ένθετες επερωτήσεις (Nested queries)
- Περιορισμοί (Constraints)

Η γλώσσα SQL

- Η SQL (Structure Query Language) είναι μια τυπική γλώσσα για τη διαχείριση σχεσιακών βάσεων δεδομένων.
- Η γλώσσα SQL είναι μία γλώσσα DML (Data Manipulation Language) καθώς παρέχει ένα σύνολο εντολών για τον χειρισμό (ανάκτηση, εισαγωγή, τροποποίηση και διαγραφή) των δεδομένων της βάσης.
- Η γλώσσα SQL είμαι μια γλώσσα DDL (Data Definition Language) καθώς παρέχει ένα σύνολο εντολών για τον ορισμό και την διαχείριση του λογικού σχήματος της βάσης.

Η γλώσσα SQL

- Εντολές χειρισμού δεδομένων (DML commands):
  - **SELECT:** χρησιμοποιείται για την ανάκτηση δεδομένων από τους πίνακες της βάσης.
  - **INSERT:** εισαγάγει δεδομένα σε έναν πίνακα.
  - **UPDATE:** μας επιτρέπει να ενημερώνουμε τα δεδομένα ενός πίνακα.
  - **DELETE:** μας δίνει την δυνατότητα να διαγράψουμε εγγραφές από έναν πίνακα.
- Εντολές ορισμού δεδομένων (DDL commands)
  - **CREATE:** χρησιμοποιείται για την δημιουργία αντικειμένων της βάσης όπως πίνακες ευρετήρια κ.λπ.
  - **ALTER:** χρησιμοποιείται για την τροποποίηση της δομής ενός πίνακα της βάσης.
  - **DROP:** χρησιμοποιείται για την διαγραφή αντικειμένων της βάσης όπως πίνακες και ευρετήρια κ.λπ.

Εντολές Χειρισμού δεδομένων (DML)

- Εντολές χειρισμού δεδομένων (DML commands):
  - **SELECT:** χρησιμοποιείται για την ανάκτηση δεδομένων από τους πίνακες της βάσης.
  - **INSERT:** εισαγάγει δεδομένα σε έναν πίνακα.
  - **UPDATE:** μας επιτρέπει να ενημερώνουμε τα δεδομένα ενός πίνακα.
  - **DELETE:** μας δίνει την δυνατότητα να διαγράψουμε εγγραφές από έναν πίνακα.

Η εντολή SELECT

- Η εντολή **SELECT** χρησιμοποιείται για την ανάκτηση δεδομένων από τους πίνακες της βάσης. Μία τυπική εντολή **select** έχει τη μορφή:

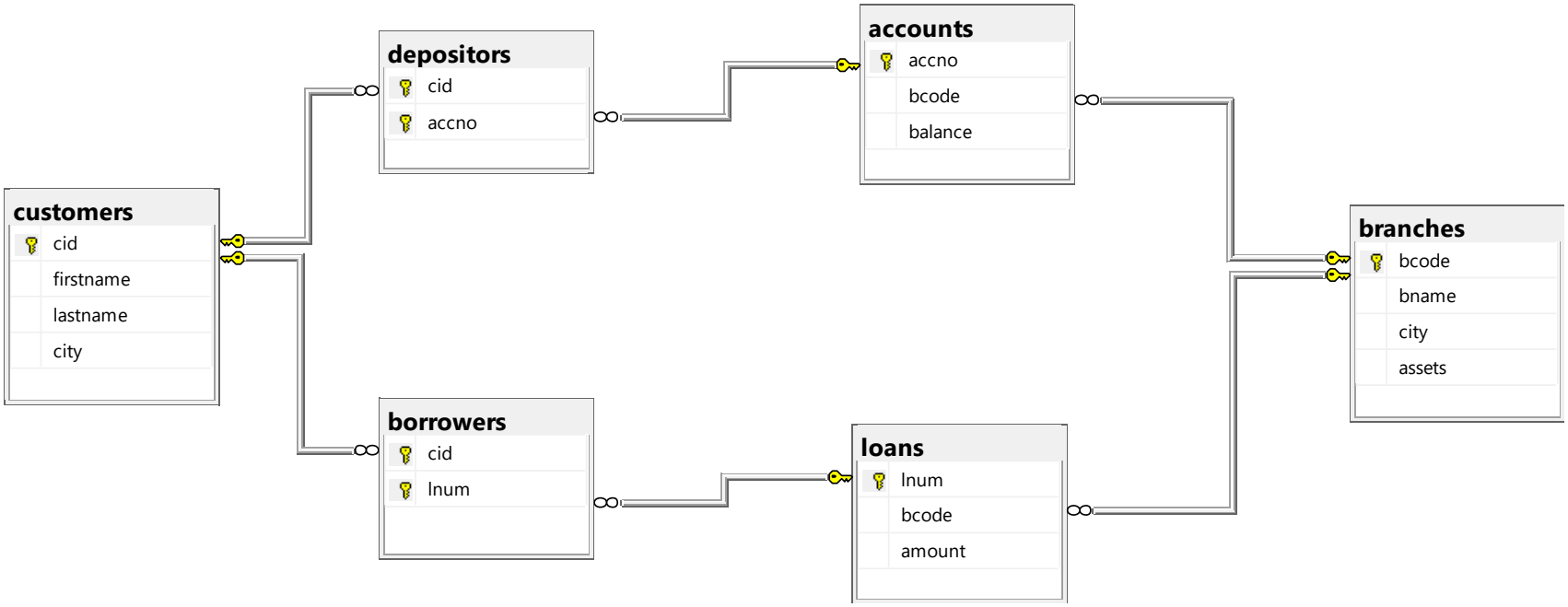
SELECT *column1, column2, …*
FROM *table_name*
WHERE *condition*

Όπου:

  - *column1, column2: στήλες(πεδία) του πίνακα*
  - *table_name: το όνομα του πίνακα*
  - *condition:* μία συνθήκη

- Μία εντολή της παραπάνω μορφής καλείται επερώτηση (query).
- Το αποτέλεσμα μιας επερώτησης είναι ένα σύνολο εγγραφών.

Σχήμα Παραδειγμάτων



Η εντολή Select - Παραδείγματα

- Εμφάνισε τα ονόματα όλων των υποκαταστημάτων της τράπεζας.

SELECT *bname*
FROM *branches*

- Εμφάνισε τα ονόματα των υποκαταστημάτων της τράπεζα που βρίσκονται στην Αθήνα.

SELECT *bname*
FROM *branches*
WHERE *city='Αθήνα'*

- Η SQL δεν κάνει διάκριση μεταξύ πεζών και κεφαλαίων στα αναγνωριστικά.

Ο προσδιοριστής DISTINCT

- Η SQL επιτρέπει την ύπαρξη διπλοτύπων στα αποτελέσματα των επερωτήσεων. Για την απαλοιφή των διπλοτύπων πρέπει να χρησιμοποιηθεί ο προσδιοριστής **DISTINCT** μετά τον όρο **SELECT.**
- Εμφάνισε όλες τις πόλεις στις οποίες η τράπεζα έχει υποκαταστήματα. Το όνομα κάθε πόλης θα εμφανιστεί μια φορά.

SELECT DISTINCT *city* **FROM** *branches*

- Ο προσδιοριστής **ALL** επιβάλει τη μη διαγραφή των διπλοτύπων. Το όνομα κάθε πόλης θα εμφανισθεί όσες φορές υπάρχει στον πίνακα branches.

SELECT ALL *city* **FROM** *branches*

- Ο προσδιοριστής **ALL** χρησιμοποιείται εξ ορισμού (by default).

Αριθμητικές εκφράσεις

- Το αστεράκι “*” μετά τον όρο **SELECT** υποδηλώνει την εμφάνιση όλων των στηλών (πεδίων) του πίνακα.

SELECT * **FROM** *loans*

- Ο όρος **SELECT** μπορεί να περιέχει αριθμητικές εκφράσεις οι οποίες σχηματίζονται με την χρήση των αριθμητικών τελεστών +, –, *, και /.
- Η επερώτηση:

SELECT *lnum, amount* * 1,01 **FROM** *loans*

εμφανίζει τον κωδικό και το ποσό κάθε δανείου προσαυξημένο κατά 1%.

Ο όρος WHERE

- Ο όρος **WHERE** χρησιμοποιείται για το φιλτράρισμα των εγγραφών βάσει μιας συνθήκης. Οι εγγραφές που θα ανακτηθούν είναι αυτές που ικανοποιούν την συνθήκη που ακολουθεί τον όρο **WHERE**

SELECT *column1, column2, …*
FROM *table_name*
WHERE *condition*

- Η παρακάτω εντολή εμφανίζει όλους τους κωδικούς των δανείων με ποσό μεγαλύτερο των 5000 Ευρώ.

SELECT *lnum*
FROM *loans*
WHERE *amount* > 5000

- Οι συνθήκες δύναται να περιέχουν συγκριτικούς τελεστές (<,>,<=, >=, =, !=), λογικούς τελεστές (and, or, not) καθώς και συνδυασμό αυτών.

Ο όρος WHERE (Συνέχεια)

- Η SQL διαθέτει και τον συγκριτικό τελεστή **BETWEEN**
- Π.χ. Εμφάνισε τον κωδικό των δανείων των οποίων το ποσό κυμαίνεται μεταξύ 5000 και 15000 ευρώ.

```
SELECT lnum
        FROM loans
  WHERE amount BETWEEN 5000 AND 15000
```

- Το ίδιο ερώτημα με χρήση λογικών τελεστών

```
SELECT lnum
        FROM loans
  WHERE amount >= 5000 AND  amount <=15000
```

Ο όρος FROM

- Ο όρος **FROM** προσδιορίζει του πίνακες που χρησιμοποιούνται στην επερώτηση.
- Βρες τον κωδικό και το ονοματεπώνυμο όλων των πελατών που διαθέτουν λογαριασμό σε οποιοδήποτε υποκατάστημα της τράπεζας.

SELECT DISTINCT customers.cid, firstname, lastname
FROM *customers, depositors*
***W*HERE** *customers.cid = depositors.cid*

Ο όρος AS - Alias for Columns

- Ο προσδιοριστής **AS** μας επιτρέπει να μετονομάσουμε μία στήλη με ένα ψευδώνυμο (Alias). Με άλλα λόγια μας δίνει την δυνατότητα να ορίζουμε ψευδώνυμα για τα ονόματα των στηλών του αποτελέσματος μιας εντολής select.

SELECT column_name **AS** alias_name **FROM** table_name

- Εμφάνισε το όνομα, το επώνυμο, τον κωδικό και το ποσό των δανείων όλων των πελατών. Μετονόμασε την στήλη lnum σε loan_number.

SELECT *firstname, lastname, borrowers.lnum* **AS** *loan_number, amount*
FROM customers,*borrowers, loans*
WHERE customers.cid=borrowers.cid **AND**
borrowers.lnum = loans.lnum

- **Προσοχή:** Αλλάζει το όνομα της στήλης στο αποτέλεσμα της επερώτησης και όχι στον πίνακα της βάσης.
- Τα ψευδώνυμα είναι ιδιαίτερα χρήσιμα για την μετονομασία των στηλών που δημιουργούνται από την χρήση συναθροιστικών συναρτήσεων σε μία εντολή SELECT.

Ο όρος AS - Alias for Tables

- Ο όρος **AS** μας επιτρέπει επίσης να χρησιμοποιούμε ψευδώνυμα για τα ονόματα των πινάκων.

SELECT column_name **FROM** table_name **AS** alias_name

- Εμφάνισε έναν κατάλογο με τα δάνεια των πελατών. Ο κατάλογος θα περιέχει το ονοματεπώνυμο των πελατών και τα στοιχεία των δανείων τους (κωδικό δανείου και ποσό).

SELECT *firstname, lastname, B.lnum , amount*
FROM customers **AS** C, *borrowers **AS** B, loans **AS** L*
WHERE C.cid=B.cid **AND** *B.lnum = L.lnum*

  - *Το C είναι ψευδώνυμο για τον πίνακα customers*
  - *Το B είναι ψευδώνυμο για τον πίνακα borrowers*
  - *Το L είναι ψευδώνυμο για τον πίνακα loans*

- **Προσοχή:** η χρήση του τελεστή **AS** είναι προαιρετική όπως φαίνεται στην ακόλουθη επερώτηση.

SELECT *firstname, lastname, B.lnum , amount*
FROM customers C, *borrowers B, loans L*
WHERE C.cid=B.cid **AND** *B.lnum = L.lnum*

Ο όρος AS - Alias for Tables

- Η απόδοση ψευδώνυμων σε ονόματα πινάκων είναι ιδιαίτερα χρήσιμη στην περίπτωση που θέλουμε να συγκρίνουμε εγγραφές του ιδίου πίνακα. Για παράδειγμα υποθέστε ότι θέλουμε να εκτελέσουμε την ακόλουθη επερώτηση

«Βρείτε τα ονόματα όλων των υποκαταστημάτων που έχουν κεφάλαια μεγαλύτερα τουλάχιστον από ένα υποκατάστημα που βρίσκεται στην Θεσσαλονίκη»

```
SELECT DISTINCT T.bname
    FROM branches AS T, branches AS S
WHERE T.assets > S.assets AND
    S.city='Θεσσαλονίκη'
```

Ο τελεστής LIKE

- Για να συγκρίνουμε συμβολοσειρές μπορούμε να χρησιμοποιήσουμε τους τελεστές σύγκρισης (=, <, >, <=, >=, !=).
- Επιπλέον η SQL υποστηρίζει την αναζήτηση και ταύτιση σχηματισμών κειμένου (pattern) με τον τελεστή **LIKE** και με την χρήση των ακόλουθων ειδικών συμβόλων γνωστών ως χαρακτήρες μπαλαντέρ (wildcards) :
  - Το σύμβολο του ποσοστού (%) το οποίο συμβολίζει μηδέν ή περισσότερους οποιουσδήποτε χαρακτήρες και είναι γνωστό ως σύμβολο απόκρυψης τμήματος κειμένου.
  - Την κάτω παύλα (_) η οποία συμβολίζει ακριβώς έναν, οποιονδήποτε, χαρακτήρα και είναι γνωστή ως σύμβολο αντικατάστασης χαρακτήρα.
- Εμφάνισε όλα τα στοιχεία των πελατών των οποίων το επώνυμο ξεκινάει με “Α”.

  SELECT * **FROM** *customers* **WHERE** *lastname* **LIKE** *‘Α%’*
- Εμφάνισε όλα τα στοιχεία των πελατών των οποίων το επώνυμο τελειώνει σε “άκης”

  SELECT * **FROM** *customers* **WHERE** *lastname* **LIKE** *‘%άκης’*
- Εμφάνισε όλα τα στοιχεία των πελατών των οποίων το επώνυμο περιέχει τουλάχιστον δύο φορές το γράμμα “α”.

  SELECT * **FROM** customers **WHERE** lastname **LIKE** ‘%α%α%’
- Βρες όλα τα ονόματα των πελατών των οποίων το επώνυμο ξεκινάει με “Κ” και έχουν μήκος 6 χαρακτήρες.

  SELECT * **FROM** customers **WHERE** lastname **LIKE ‘Κ_____’ (Το γράμμα Κ ακολουθείται από 5 κάτω παύλες).**

Χαρακτήρες Μπαλαντέρ στον SQL SERVER

Σύμβολο	Περιγραφή	Παράδειγμα
%	Αναπαριστά μηδέν ή περισσότερους χαρακτήρες	**bl%** αναπαριστά bl, black, blue, blob
_	Αναπαριστά έναν οποιονδήποτε χαρακτήρα	**h_t** αναπαριστά: hot, hat, hit
[]	Αναπαριστά οποιονδήποτε απλό χαρακτήρα που εσωκλείεται μέσα στις αγκύλες	h[oa]t αναπαριστά hot και hat, αλλά δεν αναπαριστά hit
^	Αναπαριστά οποιονδήποτε χαρακτήρα δεν εσωκλείεται στις αγκύλες.	h[^oa]t αναπαριστά hit, αλλά δεν αναπαριστά hot και hat
-	Αναπαριστά οποιονδήποτε χαρακτήρα στο συγκεκριμένο διάστημα.	c[a-b]t αναπαριστά cat και cbt

Ταξινόμηση αποτελεσμάτων (ORDER BY)

- Ο προσδιοριστής **ORDER BY** χρησιμοποιείται για την ταξινόμηση των αποτελεσμάτων σε αύξουσα (ASC) ή φθίνουσα (DESC) διάταξη.
- Εμφάνισε σε αλφαβητική σειρά με βάση το επώνυμο τα ονόματα των πελατών που έχουν λογαριασμό με υπόλοιπο πάνω από 50000.

SELECT DISTINCT lastname, firstname
FROM *customers, depositors, accounts*
WHERE *customers.cid=depositors.cid* **AND**
depositors.accno = accounts.accno ***AND*** *balance >=50000*
ORDER BY *lastname, firstname*

- Μπορούμε να ορίσουμε **DESC** για φθίνουσα ταξινόμηση.
  - Π.χ. **ORDER BY** *lastname* **DESC**
- Προκαθορισμένη διάταξη (by default) είναι η αύξουσα (ASC).

Πράξεις Συνόλων – UNION (Ένωση)

- Ο τελεστής **UNION** συνδυάζει τα αποτελέσματα δύο ή περισσότερων εντολών **SELECT.**
  - Κάθε εντολή select πρέπει να έχει τον ίδιο αριθμό πεδίων
  - Τα πεδία πρέπει να είναι του ιδίου τύπου
  - Τα πεδία κάθε εντολής select πρέπει να είναι στην ίδια σειρά.

```
SELECT column1, column2  FROM table1
   UNION
SELECT column1, column2  FROM table2;
```

- Το τελικό αποτέλεσμα είναι ένα σύνολο εγγραφών το οποίο περιλαμβάνει τις εγγραφές που επιστρέφουν και οι δύο εντολές SELECT. Δηλαδή περιέχει τις εγγραφές που επιστρέφει η πρώτη εντολή select και τις εγγραφές που επιστρέφει η δεύτερη select εντολή. Οι εγγραφές που περιέχονται και στα δύο σύνολα (κοινές εγγραφές) εμφανίζονται μία φορά. Αν θέλουμε να διατηρήσουμε τις διπλότυπες εγγραφές τότε πρέπει να χρησιμοποιήσουμε τον τελεστή **UNION ALL.**

Πράξεις Συνόλων

Βρες όλους τους πελάτες που έχουν ένα δάνειο, ένα λογαριασμό ή και τα δύο:

```
SELECT lastname,firstname
    FROM customers,depositors
  WHERE  customers.cid=depositors.cid

UNION

SELECT lastname,firstname
    FROM customers,borrowers
  WHERE  customers.cid=borrowers.cid
```

Πράξεις Συνόλων – INTERSECT (Τομή)

- Ο τελεστής **INTERSECT** επιστρέφει τα κοινά αποτελέσματα δύο ή περισσοτέρων εντολών **SELECT.**
  - Κάθε εντολή select πρέπει να έχει τον ίδιο αριθμό πεδίων
  - Τα πεδία πρέπει να είναι του ιδίου τύπου
  - Τα πεδία κάθε εντολής select πρέπει να είναι στην ίδια σειρά.

```
SELECT column1, column2  FROM table1
    INTERSECT
SELECT column1, column2  FROM table2;
```

- Το τελικό αποτέλεσμα είναι το σύνολο των εγγραφών το οποίο περιέχει τις κοινές εγγραφές που επιστρέφουν και οι δύο εντολές select. Αν μία εγγραφή υπάρχει μόνο στο αποτέλεσμα της μιας εκ των δύο εντολών select τότε δεν περιλαμβάνεται στο τελικό αποτέλεσμα.

Πράξεις Συνόλων – INTERSECT (Τομή)

Βρες όλους τους πελάτες που διαθέτουν τουλάχιστον έναν λογαριασμό και επίσης έχουν πάρει δάνειο:

```
SELECT lastname,firstname
    FROM customers,depositors
  WHERE  customers.cid=depositors.cid

INTERSECT

SELECT lastname,firstname
    FROM customers,borrowers
  WHERE  customers.cid=borrowers.cid
```

Πράξεις Συνόλων – EXCEPT (εξαίρεση)

- Ο τελεστής **EXCEPT** επιστρέφει τις εγγραφές μιας εντολής **SELECT** οι οποίες δεν περιέχονται στο αποτέλεσμα μιας άλλης εντολής **SELECT.**
  - Κάθε εντολή select πρέπει να έχει τον ίδιο αριθμό πεδίων
  - Τα πεδία πρέπει να είναι του ιδίου τύπου
  - Τα πεδία κάθε εντολής select πρέπει να είναι στην ίδια σειρά.

```
SELECT column1, column2  FROM table1
  EXCEPT
SELECT column1, column2  FROM table2;
```

- Το τελικό αποτέλεσμα είναι το σύνολο των εγγραφών που επιστρέφει η πρώτη εντολή select και δεν περιλαμβάνονται στο σύνολο των εγγραφών που επιστρέφει η δεύτερη εντολή select.
- Σε αντίθεση με τους τελεστές UNION και INTERSECT στην περίπτωση της εντολής EXCEPT η σειρά των εντολών select είναι ιδιαίτερα σημαντική.

Πράξεις Συνόλων – EXCEPT (εξαίρεση)

Βρες όλους τους πελάτες που διαθέτουν λογαριασμό και δεν έχουν πάρει δάνειο.

```
SELECT lastname,firstname
    FROM customers,depositors
  WHERE  customers.cid=depositors.cid

EXCEPT

SELECT lastname,firstname
    FROM customers,borrowers
  WHERE  customers.cid=borrowers.cid
```

Συναθροιστικές Συναρτήσεις (Aggregate functions)

- Οι παρακάτω συναρτήσεις εφαρμόζουν σε ένα σύνολο τιμών μιας στήλης και επιστρέφουν ως αποτέλεσμα μία μόνο τιμή.
  - **AVG:** μέση τιμή
  - **MIN:** ελάχιστη τιμή
  - **MAX:** μέγιστη τιμή
  - **SUM:** άθροισμα τιμών
  - **COUNT:** πλήθος τιμών

Συναθροιστικές Συναρτήσεις (Συνέχεια)

- Βρες το μέσο όρο των λογαριασμών του υποκαταστήματος "Σταδίου".

```
SELECT AVG (balance)
        FROM branches,accounts
    WHERE branches.bcode=accounts.bcode AND
            bname = 'Σταδίου'
```

- Βρες τον αριθμό των πελατών της τράπεζας.

```
SELECT COUNT (cid)
        FROM customers
```

- Βρες τον αριθμό των καταθετών της τράπεζας.

```
SELECT COUNT (distinct cid)
        FROM depositors
```

Ομαδοποίηση (GROUP BY)

- Ο όρος **GROUP BY** χρησιμοποιείται για την ομαδοποίηση των αποτελεσμάτων μιας εντολής select.
- Συνήθως χρησιμοποιείται σε συνδυασμό με τις συναθροιστικές συναρτήσεις (COUNT, SUM, AVG, MIN, MAX) για την ομαδοποίηση των αποτελεσμάτων με ένα ή περισσότερα πεδία.

SELECT *column1, function_name(column2)*
FROM table_name
WHERE condition
GROUP BY *column1*

Σημείωση: Τα πεδία που προσδιορίζονται με τον όρο **SELECT,** εκτός των συναθροιστικών συναρτήσεων, πρέπει να ακολουθούν τον προσδιοριστή **GROUP BY.**

GROUP BY: Παράδειγμα

- Βρες τον αριθμό των καταθετών ανά υποκατάστημα.

SELECT *bname,* **COUNT (DISTINCT** *cid)*
FROM branches,*depositors, accounts*
WHERE branches.bcode=accounts.bcode **AND**
depositors.accno = accounts.accno
GROUP BY *bname*

Σημείωση: Τα γνωρίσματα που προσδιορίζονται με τον όρο **SELECT,** εκτός των συναθροιστικών συναρτήσεων, πρέπει να ακολουθούν τον προσδιοριστή **GROUP BY.**

Ο όρος HAVING

- Ο όρος **HAVING** χρησιμοποιείται για να εφαρμόσουμε κάποια συνθήκη στις ομάδες που δημιουργεί ο όρος **GROUP BY**

SELECT *column1, function_name(column2)*
FROM table_name
WHERE condition
GROUP BY *column1*
HAVING *condition1*

ΠΡΟΣΟΧΗ: η συνθήκη που ακολουθεί τον όρο **HAVING** εφαρμόζεται μετά την δημιουργία των ομάδων, σε αντίθεση με τη συνθήκη που ακολουθεί τον όρο **WHERE** η οποία εφαρμόζεται πριν την δημιουργία των ομάδων. Με άλλα λόγια αν οι όροι **WHERE** και **HAVING** εμφανίζονται στην ίδια επερώτηση, η SQL εφαρμόζει πρώτα την συνθήκη που ακολουθεί τον όρο **WHERE.** Οι εγγραφές που ικανοποιούν την συνθήκη του όρου **WHERE** τοποθετούνται σε ομάδες με τη χρήση του **GROUP BY**. Στη συνέχεια η SQL εφαρμόζει τον όρο **HAVING** σε κάθε ομάδα. Αφαιρεί τις ομάδες που δεν ικανοποιούν την συνθήκη που ακολουθεί τον όρο **HAVING.**

Ο όρος HAVING - Παράδειγμα

- Βρες τα ονόματα των υποκαταστημάτων για τα οποία ο μέσος όρος των καταθέσεων είναι μεγαλύτερος από 10000

SELECT *bname,* ***AVG*** **(**balance*)*
FROM branches,*accounts*
WHERE branches.bcode=accounts.bcode
GROUP BY *bname*
HAVING AVG*(balance) > 10000*

Τιμές NULL

- Η SQL διαθέτει μία ειδική τιμή η οποία ονομάζεται **NULL (**το απόλυτο τίποτα).
- Η ειδική τιμή NULL χρησιμοποιείτε σε περιπτώσεις όπου:
  - Υπάρχει τιμή, αλλά δεν την ξέρουμε (π.χ., δεν γνωρίζουμε το τηλέφωνο ενός πελάτη),
  - Δεν ξέρουμε αν υπάρχει τιμή (π.χ. δεν ξέρουμε αν ένας συγκεκριμένος πελάτης έχει κινητό τηλέφωνο)
  - Η τιμή δεν ορίζεται (π.χ. το πεδίο επώνυμο_συζύγου δεν ορίζεται για ένα συγκεκριμένο πελάτη που δεν έχει σύζυγο)
- Αν ένα πεδίο (στήλη) ενός πίνακα είναι προαιρετικό, μπορούμε να εισαγάγουμε ή να τροποποιήσουμε μία εγγραφή δίχως να προσδιορίσουμε μία τιμή για το συγκεκριμένο πεδίο. Σε αυτή την περίπτωση το πεδίο θα πάρει την τιμή NULL.
- **ΠΡΟΣΟΧΗ:** Η τιμή NULL είναι διαφορετική από την τιμή μηδέν ή τον χαρακτήρα του κενού διαστήματος “ “ (space). Με άλλα λόγια ένα αριθμητικό πεδίο το οποίο έχει την τιμή 0 (μηδέν), ή ένα πεδίο χαρακτήρων που περιέχει έναν ή περισσότερα κενά διαστήματα δεν έχουν την τιμή NULL.

Τιμές NULL

- Η SQL διαθέτει τον ειδικό τελεστή **IS NULL** για να ελέγχει την ύπαρξη της τιμής **NULL.**
- Π.χ. βρες όλους τους κωδικούς των δανείων που έχουν στο πεδίο *amount την τιμή null.*

```
SELECT lnum
    FROM loans
 WHERE amount IS NULL
```

- Το αποτέλεσμα κάθε αριθμητικής έκφρασης που περιέχει την τιμή *NULL* είναι *NULL*
  - Π.χ. 5 + NULL = NULL

Τιμές NULL

- Κάθε σύγκριση με τιμές *null* επιστρέφει *unknown*
  - *Π.χ. 5 < NULL OR NULL <> NULL OR NULL = NULL*
- Λογικοί τελεστές με χρήστη της τιμής *unknown*:
  - OR: (*unknown* **OR** *true*) = *true*, (*unknown* **OR** *false*) = *unknown*, (*unknown* **or** *unknown*) = *unknown*
  - AND: *(true* **AND** *unknown) = unknown, (false* **AND** *unknown) = false, (unknown* **AND** *unknown) = unknown*
  - NOT*: (***NOT** *unknown) = unknown*
- Το αποτέλεσμα της συνθήκης του όρου **WHERE** είναι false αν αποτιμάται σε *unknown*

Τιμές NULL

- Υπολόγισε το συνολικό ποσό δανείων

  SELECT SUM (*amount)*
  FROM *loans*

  – Η παραπάνω εντολή αγνοεί όλες τις τιμές NULL.

- Όλες οι συναθροιστικές συναρτήσεις, εκτός από την **COUNT(*),** αγνοούν τις τιμές **NULL** των εγγραφών στα πεδία που δέχονται ως όρισμα.

Ένθετες επερωτήσεις

- Ένθετη επερώτηση καλείτε μία επερώτηση **SELECT-FROM-WHERE** η οποία εσωκλείεται σε μία άλλη επερώτηση.
- Για την δημιουργία ένθετων επερωτήσεων SELECT χρησιμοποιούμε τους τελεστές **IN** και **NOT IN**.

SELECT column_name
FROM *table_name*
WHERE *column_name* ***IN*** *(**SELECT** statement)*

SELECT column_name
FROM *table_name*
WHERE *column_name* ***NOT IN****(**SELECT** statement)*

Παραδείγματα ένθετων επερωτήσεων

- Βρες όλους τους πελάτες που διαθέτουν λογαριασμό και έχουν πάρει δάνειο.

```
SELECT  DISTINCT lastname,firstname
     FROM customers,depositors
   WHERE  customers.cid=depositors.cid AND
            customers.cid IN (select cid from borrowers)
```

- Βρες όλους τους πελάτες που διαθέτουν λογαριασμό και δεν έχουν πάρει δάνειο.

```
SELECT DISTINCT lastname,firstname
     FROM customers,depositors
   WHERE  customers.cid=depositors.cid AND
            customers.cid NOT IN (SELECT cid FROM borrowers)
```

Σύγκριση Συνόλων

- Η SQL παρέχει τους τελεστές **SOME**, **ANY** και **ALL** για την σύγκριση συνόλων ένθετων επερωτήσεων. Ο τελεστής **ANY** είναι συνώνυμο του τελεστή **SOME.**

SELECT column_name
FROM *table_name*
WHERE *column_name συγκριτικός_τελεστής* {***SOME/ALL***} (subquery)

- Όπου συγκριτικός τελεστές ένας εκ των >, <, >=, <=, =, !=

Παράδειγμα χρήσης του τελεστή SOME

- Βρείτε όλα τα υποκαταστήματα με κεφάλαια μεγαλύτερα τουλάχιστον από ένα υποκατάστημα που βρίσκεται στην Πάτρα.

SELECT *bname*
FROM *branches*
WHERE *assets* > **SOME**
(SELECT *assets*
FROM *branches*
WHERE *city =‘Πάτρα’***)**

Ο τελεστής SOME

Εκφράσεις με τον τελεστή **SOME**

(5< **SOME** {0, 5, 6}) = true

Διότι η τιμή 5 είναι μικρότερη από τουλάχιστον μια τιμή (την τιμή 6.)

(5< **SOME** {0, 5}) = false

Διότι η τιμή 5 δεν είναι μικρότερη από τουλάχιστον μια τιμή.

(5 = **SOME** {0, 5}) = true

Διότι υπάρχει η τιμή 5.

(5 ≠ **SOME** {0, 5}) = true (επειδή 0 ≠ 5)

Διότι υπάρχει μια τουλάχιστον τιμή διαφορετική από την τιμή 5.

Παράδειγμα χρήσης του τελεστή ALL

- Βρες τα ονόματα των υποκαταστημάτων με κεφάλαια μεγαλύτερα από όλα τα υποκαταστήματα της Πάτρας.

SELECT *bname*
FROM *branches*
WHERE *assets* > ***ALL***
(SELECT *assets*
FROM *branches*
WHERE *city* = ‘Πάτρα’)

Ο τελεστής All

Εκφράσεις με τον τελεστή **ALL**

(5< **ALL** | 0 / 5 / 6 |) = false

Διότι η τιμή 5 δεν είναι μικρότερη όλες τις τιμές (0,5 και 6).

(5< **ALL** | 6 / 10 |) = true

Διότι η τιμή 5 είναι μικρότερη όλες τις τιμές (6,10)

(5 = **ALL** | 4 / 5 |) = false

Διότι η τιμή 5 δεν είναι ίση με όλες τις τιμές.

(5 ≠ **ALL** | 4 / 6 |) = true

Διότι η τιμή 5 είναι διαφορετική τόσο από την τιμή 4 όσο και από την τιμή 6.

Σύζευξη Πινάκων (JOIN)

- Η λειτουργία της σύζευξης (JOIN) συνδυάζει τις εγγραφές δύο η περισσότερων πινάκων με βάση μια κοινή στήλη. Υπάρχουν οι παρακάτω τύποι σύζευξης:
  - **(INNER) JOIN**
  - **LEFT (OUTER) JOIN**
  - **RIGHT (OUTER) JOIN**
  - **FULL (OUTER) JOIN**
  - **CROSS JOIN (Καρτεσιανό γινόμενο)**
- Οι όροι που είναι μέσα σε παρένθεση είναι προαιρετικοί. Αυτό σημαίνει ότι:
  - Το **INNER JOIN** είναι ισοδύναμο με το **JOIN**
  - Το **LEFT OUTER JOIN** είναι ισοδύναμο με το **LEFT JOIN**
  - Το **LEFT OUTER JOIN** είναι ισοδύναμο με το **LEFT JOIN**
  - Το **RIGHT OUTER JOIN** είναι ισοδύναμο με το **RIGHT JOIN**
  - Το **FULL OUTER JOIN** είναι ισοδύναμο με το **FULL JOIN**

INNER JOIN (ή JOIN)

accounts

accno	balance	bcode
A900	1000	100
A905	50000	150
A925	6000	700
A907	5000	800

branches

bcode	bname	city	assets
100	Σταδίου	Αθήνα	10000000
150	Πατησίων	Αθήνα	700000
250	Τσιμισκή	Θεσσαλονίκη	500000
350	Αμαλίας	Πάτρα	400000

***SELECT** * **FROM** accounts* **INNER JOIN** *branches* **ON** *accounts.bcode = branches.bcode*

- Το αποτέλεσμα της παραπάνω εντολής αποτελείται από τις εγγραφές των δύο πινάκων που έχουν την ίδια τιμή στο κοινή στήλη **bcode** με βάση την οποία γίνεται η σύζευξη των πινάκων

accno	balance	bcode	bcode	bname	city	assets
A900	1000	100	100	Σταδίου	Αθήνα	10000000
A905	50000	150	150	Πατησίων	Αθήνα	700000

- Το ίδιο αποτέλεσμα μπορούμε να πάρουμε και με την ακόλουθη εντολή στην οποία αντί για το **INNER JOIN** χρησιμοποιούμε το **WHERE**

***SELECT** * **FROM** accounts, branches* **WHERE** *accounts.bcode = branches.bcode*

LEFT JOIN (ή LEFT OUTER JOIN)

accounts

accno	balance	bcode
A900	1000	100
A905	50000	150
A925	6000	700
A907	5000	800

branches

bcode	bname	city	assets
100	Σταδίου	Αθήνα	10000000
150	Πατησίων	Αθήνα	700000
250	Τσιμισκή	Θεσσαλονίκη	500000
350	Αμαλίας	Πάτρα	400000

***SELECT** * **FROM** accounts* **LEFT JOIN** *branches* **ON** *accounts.bcode = branches.bcode*

- Το αποτέλεσμα της παραπάνω εντολής αποτελείται από τις εγγραφές των δύο πινάκων που έχουν την ίδια τιμή στο κοινή στήλη **bcode,** καθώς επίσης και τις εγγραφές του πίνακα **accounts** που δεν πληρούν την συνθήκη της σύζευξης. Ο πίνακας accounts βρίσκεται αριστερά του τελεστή **LEFT JOIN.**

accno	balance	bcode	bcode	bname	city	assets
A900	1000	100	100	Σταδίου	Αθήνα	10000000
A905	50000	150	150	Πατησίων	Αθήνα	700000
A925	6000	700	null	null	null	null
A907	5000	800	null	null	null	null

RIGHT JOIN (ή RIGHT OUTER JOIN)

accounts

accno	balance	bcode
A900	1000	100
A905	50000	150
A925	6000	700
A907	5000	800

branches

bcode	bname	city	assets
100	Σταδίου	Αθήνα	10000000
150	Πατησίων	Αθήνα	700000
250	Τσιμισκή	Θεσσαλονίκη	500000
350	Αμαλίας	Πάτρα	400000

SELECT *** FROM*** *accounts* **RIGHT JOIN** *branches* **ON** *accounts.bcode = branches.bcode*

- Το αποτέλεσμα της παραπάνω εντολής αποτελείται από τις εγγραφές των δύο πινάκων οι οποίες έχουν την ίδια τιμή στο κοινή στήλη **bcode,** καθώς επίσης και τις εγγραφές του πίνακα **branches** που δεν πληρούν την συνθήκη της σύζευξης. Ο πίνακας branches βρίσκεται δεξιά του τελεστή **RIGHT JOIN.**

accno	balance	bcode	bcode	bname	city	assets
A900	1000	100	100	Σταδίου	Αθήνα	10000000
A905	50000	150	150	Πατησίων	Αθήνα	700000
null	null	null	250	Τσιμισκή	Θεσσαλονίκη	500000
null	null	null	350	Αμαλίας	Πάτρα	400000

FULL JOIN (ή FULL OUTER JOIN)

accounts

accno	balance	bcode
A900	1000	100
A905	50000	150
A925	6000	700
A907	5000	800

branches

bcode	bname	city	assets
100	Σταδίου	Αθήνα	10000000
150	Πατησίων	Αθήνα	700000
250	Τσιμισκή	Θεσσαλονίκη	500000
350	Αμαλίας	Πάτρα	400000

SELECT *** ***FROM*** *accounts* **FULL JOIN** *branches* **ON** *accounts.bcode = branches.bcode*

- Ο τύπος σύζευξης **FULL JOIN** συνδυάζει τα αποτελέσματα των τύπων σύζευξης **LEFT JOIN** και **RIGHT JOIN.**

accno	balane	bcode	bcode	bname	city	assets
A900	1000	100	100	Σταδίου	Αθήνα	10000000
A905	50000	150	150	Πατησίων	Αθήνα	700000
A925	6000	700	null	null	null	null
A907	5000	800	null	null	null	Null
null	null	null	250	Τσιμισκή	Θεσσαλονίκη	500000
null	null	null	350	Αμαλίας	Πάτρα	400000

Καρτεσιανό γινόμενο

accounts

accno	balance	bcode
A900	1000	100
A905	50000	150
A925	6000	700
A907	5000	800

branches

bcode	bname	city	assets
100	Σταδίου	Αθήνα	10000000
150	Πατησίων	Αθήνα	700000
250	Τσιμισκή	Θεσσαλονίκη	500000
350	Αμαλίας	Πάτρα	400000

***SELECT * FROM** accounts,branches*

- Η παραπάνω εντολή επιστρέψει 16 (4χ4) εγγραφές. Λειτουργεί ως εξής: Κάθε εγγραφή του πίνακα **accounts** συνδυάζεται με κάθε εγγραφή του πίνακα **branches**. Το αποτέλεσμα είναι γνωστό ως **καρτεσιανό γινόμενο** των πινάκων. **Προσέξτε ότι δεν έχουμε ορίσει κάποιον τύπο σύζευξης στην εντολή select**

- Το καρτεσιανό γινόμενο των πινάκων μπορούμε να το δημιουργήσουμε επίσης προσδιορίζοντας τον τύπο σύζευξης **CROSS JOIN.**

***SELECT * FROM** accounts **CROSS JOIN** branches*

Καρτεσιανό γινόμενο

accounts

accno	balance	bcode
A900	1000	100
A905	50000	150
A925	6000	700
A907	5000	800

branches

bcode	bname	city	assets
100	Σταδίου	Αθήνα	10000000
150	Πατησίων	Αθήνα	700000
250	Τσιμισκή	Θεσσαλονίκη	500000
350	Αμαλίας	Πάτρα	400000

***SELECT * FROM** accounts,branches ή **SELECT * FROM** accounts **CROSS JOIN** branches*

ACCNO	BALANCE	BCODE	BCODE	BNAME	CITY	ASSETS
A900	1000	100	100	Σταδίου	Αθήνα	10000000
A905	50000	150	100	Σταδίου	Αθήνα	10000000
A925	6000	700	100	Σταδίου	Αθήνα	10000000
A907	5000	800	100	Σταδίου	Αθήνα	10000000
A900	1000	100	150	Πατησίων	Αθήνα	700000
A905	50000	150	150	Πατησίων	Αθήνα	700000
A925	6000	700	150	Πατησίων	Αθήνα	700000
A907	5000	800	150	Πατησίων	Αθήνα	700000
A900	1000	100	250	Τσιμισκή	Θεσσαλονίκη	500000
A905	50000	150	250	Τσιμισκή	Θεσσαλονίκη	500000
A925	6000	700	250	Τσιμισκή	Θεσσαλονίκη	500000
A907	5000	800	250	Τσιμισκή	Θεσσαλονίκη	500000
A900	1000	100	350	Αμαλίας	Πάτρα	400000
A905	50000	150	350	Αμαλίας	Πάτρα	400000
A925	6000	700	350	Αμαλίας	Πάτρα	400000
A907	5000	800	350	Αμαλίας	Πάτρα	400000

Εισαγωγή εγγραφών – Η εντολή INSERT

- Η εντολή INSERT εισαγάγει (προσθέτει) μία νέα εγγραφή σε έναν πίνακα.

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

ή

INSERT INTO table_name (column1, column2, column3,...columnN)

- Πρόσθεσε μία νέα εγγραφή στον πίνακα **customers**

INSERT INTO customers **VALUES** (1, 'Μάριος', 'Αβραμίδης', 'Αθήνα')

ή ισοδύναμα:

INSERT INTO customers (cid, firstname, lastname, city)
VALUES (1, 'Μάριος', 'Αβραμίδης', 'Αθήνα')

- Πρόσθεσε μια νέα εγγραφή στον πίνακα accounts με την τιμή null στο πεδίο balance

INSERT INTO *accounts (accno, bcode, balance)*
VALUES ('A100',2, *null*)

Εισαγωγή εγγραφών σε πίνακα με επιλογή

- Έχουμε την δυνατότητα να εισαγάγουμε εγγραφές σε έναν πίνακα τις οποίες μπορούμε να επιλέξουμε από έναν άλλον πίνακα.

```
INSERT INTO table1  (column1, column2, ... columnN)
      SELECT  column1, column2, ...columnN
          FROM table2
      WHERE condition
```

- Δώσε δώρο σε όλους τους πελάτες που έχουν δάνειο στο υποκατάστημα "Πατησίων" ένα λογαριασμό καταθέσεων με υπόλοιπο 500 ευρώ. Ο αριθμός δανείου θα αποτελέσει τον αριθμό του νέου λογαριασμού για κάθε πελάτη.

```
INSERT INTO accounts
      SELECT lnum, loans.bcode,  500
        FROM loans, branches
      WHERE loans.bcode=branches.bcode AND bname='Πατησίων'

INSERT INTO depositors
      SELECT cid, lnum
        FROM loans, borrowers,branches
      WHERE loans.lnum=borrowers.lnum AND
            loans.bcode=branches.bcode AND bname='Πατησίων'
```

- Η πρόταση **SELECT FROM WHERE** αποτιμάται ολοκληρωτικά πρίν τα αποτελέσματά της καταχωρηθούν στην αντίστοιχη σχέση.

Διαγραφή εγγραφών – Η εντολή DELETE

- Μπορούμε να διαγράψουμε εγγραφές από έναν πίνακα με την εντολή **DELETE**

  DELETE FROM table_name **WHERE** condition

- Διέγραψε τον πελάτη με κωδικό 10.

  DELETE FROM *customers* **WHERE** *cid=10*

Διαγραφή εγγραφών – Η εντολή DELETE

- Διέγραψε όλες τις εγγραφές των λογαριασμών με υπόλοιπο μικρότερο του μέσου όρου της τράπεζας.

DELETE FROM *accounts*
WHERE *balance* < (**SELECT AVG** *(balance)* **FROM** *accounts)*

- Πρόβλημα: κατά την διαγραφή των γραφών ο μέσος όρος αλλάζει

Λύση στην SQL:

1. Πρώτα υπολογίζεται ο μέσος όρος **AVG** και εντοπίζονται οι εγγραφές προς διαγραφή.
2. Στη συνέχεια διαγράφονται οι παραπάνω εγγραφές δίχως νέο υπολογισμό του μέσου όρου.

Διαγραφή εγγραφών – Η εντολή DELETE

- Διέγραψε όλους τους λογαριασμούς των υποκαταστημάτων της Θεσσαλονίκης.

```
DELETE FROM depositors
 WHERE accno IN
              (SELECT accno
                    FROM branches, accounts
                  WHERE accounts.bcode=branches.bcode
                   AND city ='Θεσσαλονίκη')

DELETE FROM accounts
  WHERE bcode IN (SELECT bcode
                        FROM branches
                      WHERE city ='Θεσσαλονίκη')
```

Ενημέρωση εγγραφών – Η εντολή UPDATE

- Με την εντολή **UPDATE** μπορούμε να ενημερώσουμε τις εγγραφές ενός πίνακα.

```
UPDATE table_name
   SET column1 = value1, column2 = value2, ...
 WHERE  condition
```

- Η παρακάτω εντολή ενημερώνει το πεδίο city, της εγγραφής πελάτη με κωδικό 10, με την τιμή ‘Αθήνα’

```
UPDATE customers
    SET city=‘Αθήνα’
 WHERE cid=10
```

Ενημέρωση εγγραφών – Η εντολή UPDATE

- Δώσε αύξηση 6% σε όλους τους λογαριασμούς με υπόλοιπο μεγαλύτερο του 10000 ευρώ. Οι υπόλοιποι λογαριασμοί θα πάρουν αύξηση 5%.
  - Απαιτούνται δύο εντολές **update:**

```
UPDATE accounts
  SET balance = balance * 1.06
WHERE balance > 10000

UPDATE accounts
  SET balance = balance * 1.05
WHERE balance <= 10000
```

  - Η σειρά είναι σημαντική
  - Μπορεί να γραφεί καλύτερα με χρήση της εντολής **case**

Η εντολή CASE

- Το ίδιο ερώτημα με το προηγούμενο αλλά με χρήση της εντολής **CASE**

```
UPDATE accounts
  SET balance =  CASE
      WHEN balance <= 10000 THEN balance *1,05
      ELSE   balance * 1,06
  END
```

Εντολές ορισμού δεδομένων (DDL)

- Εντολές ορισμού δεδομένων (DDL commands)
  - **CREATE:** χρησιμοποιείται για την δημιουργία αντικειμένων της βάσης όπως πίνακες ευρετήρια κ.λπ.
  - **ALTER:** χρησιμοποιείται για την τροποποίηση της δομής ενός πίνακα της βάσης.
  - **DROP:** χρησιμοποιείται για την διαγραφή αντικειμένων της βάσης όπως πίνακες και ευρετήρια κ.λπ.

Η εντολή CREATE TABLE

- Η εντολή **CREATE TABLE** δημιουργεί έναν νέο πίνακα στην βάση δεδομένων.

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

- Παράδειγμα: Δημιουργία του πίνακα branches

```
CREATE TABLE branches
(
    bcode INT
    bname VARCHAR(30),
    city VARCHAR(30),
    assets NUMERIC(18, 0)
)
```

Η εντολή ALTER TABLE

- Με την εντολή **ALTER TABLE** μπορούμε να προσθέσουμε, να διαγράψουμε ή να τροποποιήσουμε τις στήλες ενός πίνακα.
- Προσθήκη στήλης

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

- Διαγραφή στήλης

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

- Τροποποίηση στήλης

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

Η εντολή ALTER TABLE - Παραδείγματα

- Προσθήκη της στήλης birthDate στον πίνακα customers

```
alter table customers
   add birthdate date
```

- Τροποποίηση στήλης birthdate του πίνακα customers

```
alter table table_name
    alter column birthdate int
```

  – Στην my SQL και την Oracle αντί του όρου **alter column** χρησιμοποιείται ο όρος **modify column**

- Διαγραφή της στήλης email του πίνακα customers

```
alter table customers
    drop column birthdate
```

Η εντολή DROP TABLE

- Η εντολή **DROP TABLE** διαγράφει έναν πίνακα από την βάση δεδομένων.

  DROP TABLE table_name

- Διαγραφή του πίνακα customers

  DROP TABLE customers

Περιορισμοί (Constraints)

- Μπορούμε να ορίσουμε περιορισμούς κατά την δημιουργία ενός πίνακα με την εντολή **CREATE TABLE,** η μετά την δημιουργία του πίνακα με την εντολή **ALTER TABLE.**

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    ……
    )
```

- Με τους περιορισμούς μπορούμε να ορίσουμε κανόνες για τα δεδομένα ενός πίνακα. Οι ακόλουθοι περιορισμοί χρησιμοποιούνται συχνά στην SQL:
  - **NOT NULL**: διασφαλίζει ότι μια στήλη δεν μπορεί να δεχτεί τιμές NULL
  - **UNIQUE**: διασφαλίζει ότι όλες οι τιμές μιας στήλης είναι μοναδικές.
  - **PRIMARY KEY**: Συνδυασμός των περιορισμών Not null και Unique. Προσδιορίζει μοναδικά κάθε εγγραφή ενός πίνακα.
  - **FOREIGN KEY:** Διασφαλίζει την ακεραιότητα των συνδέσμων μεταξύ πινάκων (αναφορική ακεραιότητα).
  - **CHECK:** Διασφαλίζει ότι οι τιμές μιας στήλης ικανοποιούν μια συγκεκριμένη συνθήκη.

Περιορισμοί (Constraints)

Παράδειγμα: Όρισε το πεδίο **bcode** πρωτεύον κλειδί του πίνακα branches, διασφάλισε ότι τα πεδία **bname** και **city** δεν δέχονται τιμές null, και ότι το πεδίο **assets** δεν δέχεται αρνητικούς αριθμούς.

```
CREATE TABLE branches
(
    bcode INT,
    bname VARCHAR(30) NOT NULL,
    city VARCHAR(30) NOT NULL,
    assets NUMERIC(18, 0) NOT NULL,
        PRIMARY KEY (bcode),
        CHECK (assets)>=0
)
```

Περιορισμοί (Constraints)

Παράδειγμα: Όρισε το πεδίο **accno** πρωτεύον κλειδί του πίνακα accounts, διασφάλισε ότι τα πεδία **bcode** και **balance** δεν δέχονται τιμές null. Δημιούργησε μια σχέση πρωτεύοντος-ξένου κλειδιού μεταξύ των πινάκων account και branches. Το πεδίο **bcode** του πίνακα account είναι ξένο κλειδί το οποίο αναφέρεται στο πρωτεύον κλειδί **bcode** του πίνακα branches.

```
CREATE TABLE accounts (
    accno   VARCHAR(10),
    bcode   INT  NOT NULL,
    balance NUMERIC(18, 0) NOT NULL,
            PRIMARY KEY (accno),
            CONSTRAINT  fk_account FOREIGN KEY (bcode)
                                   REFERENCES   branches(bcode)
)
```

Τύποι Δεδομένων (SQL Server)

String Data Types

Data type	Description	Max size	Storage
CHAR(N)	Fixed width character string	8,000 characters	Defined width
VARCHAR(N)	Variable width character string	8,000 characters	2 bytes + number of chars
VARCHAR(MAX)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
TEXT	Variable width character string	2GB of text data	4 bytes + number of chars
NCHAR	Fixed width Unicode string	4,000 characters	Defined width x 2
NVARCHAR	Variable width Unicode string	4,000 characters	
NVARCHAR(MAX)	Variable width Unicode string	536,870,912 characters	
NTEXT	Variable width Unicode string	2GB of text data	
BINARY(N)	Fixed width binary string	8,000 bytes	
VARBINARY	Variable width binary string	8,000 bytes	
VARBINARY(MAX)	Variable width binary string	2GB	
IMAGE	Variable width binary string	2GB	

Τύποι Δεδομένων (SQL Server)

Numeric Data Types (1)

Data type	Description	Storage
BIT	Integer that can be 0, 1, or NULL	
TINYINT	Allows whole numbers from 0 to 255	1 byte
SMALLINT	Allows whole numbers between -32,768 and 32,767	2 bytes
INT	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
BIGINT	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
DECIMAL(P,S)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes

Τύποι Δεδομένων (SQL Server)

Numeric Data Types (2)

Data type	Description	Storage
NUMERIC(P,S)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
SMALLMONEY	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
MONEY	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
FLOAT(N)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
REAL	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

Τύποι Δεδομένων (SQL Server)

Date and Time Data Types

Data type	Description	Storage
DATETIME	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
DATETIME2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
SMALLDATETIME	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
DATE	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
TIME	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
DATETIMEOFFSET	The same as datetime2 with the addition of a time zone offset	8-10 bytes
TIMESTAMP	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable	

Η εντολή CREATE DATABASE

- Η εντολή της SQL **CREATE DATABASE** χρησιμοποιείται για την δημιουργία μιας βάσης δεδομένων

  CREATE DATABASE database_name

- Στον SQL server μπορούμε κατά την δημιουργία να ορίσουμε και το collation της βάσης ως εξής:

  CREATE DATABASE database_name **COLLATE** collation_name

- Για παράδειγμα η παρακάτω εντολή δημιουργεί την βάση δεδομένων “bank” με collation το “Greek_CI_AS”

  CREATE DATABASE bank **COLLATE** Greek_CI_AS

- Σε περίπτωση που έχουμε πολλές βάσεις δεδομένων η εντολή use χρησιμοποιείται για να επιλέξουμε αυτήν με την οποία θέλουμε να εργαστούμε.

  USE database_name

Sql Server: Η εντολή GO

- Η εντολή **GO** δηλώνει το τέλος μιας δέσμης (batch) εντολών SQL. Όταν η λέξη **GO** ακολουθείτε από έναν θετικό αριθμό έστω n τότε η δέσμη εντολών θα εκτελεστεί n φορές.

  GO [count]

- Για παράδειγμα η παρακάτω δέσμη εντολών (δηλαδή οι δύο εντολές select) θα εκτελεστούν 5 φορές:

  SELECT * **FROM** customers
  SELECT * **FROM** branches
  GO 5

– Η εντολή **GO** δεν είναι εντολή της SQL. Είναι μια εντολή του SQL server η οποία διευκολύνει την αναγνωσιμότητα και την εκτέλεση δεσμών εντολών (batches) και σεναρίων (scripts).

Ο χαρακτήρας “; “ (semicolon)

- Ο χαρακτήρας **;** (ελληνικό ερωτηματικό, semicolon) δηλώνει το τέλος μιας εντολής SQL. Αν και στον SQL SERVER η χρήση του ελληνικού ερωτηματικού δεν είναι υποχρεωτική, **συνίσταται να το χρησιμοποιείται πάντα** καθώς το πρότυπο ANSI SQL ορίζει ρητά τη χρήση του συγκεκριμένου χαρακτήρα μετά από κάθε εντολή SQL.

Βιβλιογραφία

- Ramakrishnan and Gehrke (2012) Συστήματα Διαχείρισης Βάσεων Δεδομένων. Τρίτη έκδοση, Εκδόσεις Τζιόλα.
- Silberschatz, Korth, Sudarsham (2001). Database system concepts. Fourth Edition, McGraw-Hill.
- Γιαννακουδάκης, Εμμανουήλ Ι. (2014) Βάσεις δεδομένων: θεωρία και πράξη, Αθήνα: Μπένος.