SQL: Συναρτήσεις Συνάθροισης

- Συναρτήσεις Συνάθροισης (Aggregate Functions)
- Εφαρμόζονται πάνω σε σύνολα τιμών γνωρισμάτων.
- **count, max, min, avg, sum**
- Περιορισμοί:
  - η συνάρτηση **count** μπορεί να εφαρμοστεί σε γνωρίσματα οποιουδήποτε τύπου
  - οι συναρτήσεις **avg** και **sum** εφαρμόζονται μόνο σε γνωρίσματα αριθμητικών τύπων
  - οι συναρτήσεις **min** και **max** εφαρμόζονται σε γνωρίσματα αριθμητικών ή αλφαριθμητικών τύπων

SQL: Συναρτήσεις Συνάθροισης

Orders(orderno, month,cid,aid,pid,qty,amt)

Agents(aid, aname,city,percent)

Customers(cid, cname,city,discnt)

- Παραδείγματα:
  1. Υπολογίστε το συνολικό ποσό όλων των παραγγελιών

     ```
     select sum(amt) from orders;
     ```
  2. Υπολογίστε τη συνολική ποσότητα του προϊόντος p03 που έχει παραγγελθεί.

     ```
     select sum(qty) as TOTAL from orders where pid='p03';
     ```
  3. Βρείτε το συνολικό αριθμό πελατών.

     ```
     select count(cid) from customers;
     ```

     ή ισοδύναμα: `select count(*) from customers;`

     Οι κενές τιμές δεν μετρούνται. Οι δύο εκφράσεις δίνουν την ίδια απάντηση γιατί δεν επιτρέπονται κενές τιμές στο γνώρισμα cid.

SQL: Συναρτήσεις Συνάθροισης

Orders(orderno, month,cid,aid,pid,qty,amt)

Agents(aid, aname,city,percent)

Customers(cid, cname,city,discnt)

4. Βρείτε το συνολικό αριθμό πόλεων όπου υπάρχουν πελάτες.

```
select count (distinct city) from customers;
```

5. Βρείτε τα ids των πελατών των οποίων η έκπτωση είναι μικρότερη από τη μέγιστη έκπτωση.

```
select cid from customers where discnt <
max(discnt);
```

Η έκφραση είναι λανθασμένη! Συγκρίσεις με συναρτήσεις συνάθροισης επιτρέπονται μόνο όταν η συνάρτηση επιστρέφεται από subselect.

Η ορθή έκφραση είναι:

```
select cid from customers where discnt <
(select max(discnt)from customers);
```

SQL: Κενές τιμές

- Κενές τιμές στην SQL:
  - Μια κενή τιμή είναι μια ειδική σταθερά η οποία αναπαριστά μια τιμή η οποία είτε δεν είναι γνωστή είτε δεν έχει νόημα για ένα συγκεκριμένο στιγμιότυπο
  - Τα περισσότερα ΣΔΒΔ δε διαφοροποιούν τις δύο ερμηνείες των κενών τιμών
  - Κενές τιμές μπορούν να εισαχθούν με την εντολή insert
  - Παράδειγμα: Ένας νέος πελάτης εισάγεται στη σχέση customers αλλά δεν είναι γνωστή η τιμή του γνωρίσματος discnt.

```
insert into customers (cid, cname, city) values
   ('c007', 'James Bond', 'London');
```

Η τιμή στο γνώρισμα discnt θα είναι null.

Customers(cid, cname,city,discnt)

SQL: Κενές τιμές

- Κενές τιμές στην SQL:
  - Το αποτέλεσμα μιας σύγκρισης με μια κενή τιμή είναι unknown (ούτε true, ούτε false).
  - Παράδειγμα: η ερώτηση

    ```
    select * from customers where discnt <=10 or
    discnt >10;
    ```

    δε θα επιστρέψει την πλειάδα με cid='c007' του προηγούμενου παραδείγματος
  - Κενές τιμές μπορούν να ανακτηθούν με χρήση του κατηγορήματος `is null`
  - Παράδειγμα: `select * from customers where discnt is null;`

SQL: Κενές τιμές

- Κενές τιμές στην SQL:
  - Οι κενές τιμές **δε** συμμετέχουν στον υπολογισμό συναρτήσεων συνάθροισης
  - Παράδειγμα: Βρείτε τη μέση έκπτωση των πελατών

    Η ερώτηση **`select avg(discnt) from customers;`**

    δε θα συμπεριλάβει την πλειάδα με cid='c007'
  - Κενές τιμές μπορούν να επιστραφούν σαν το αποτέλεσμα συναρτήσεων συνάθροισης αν αυτές υπολογιστούν πάνω στο κενό σύνολο:
    - Οι συναρτήσεις **`avg, sum, max, min`** επιστρέφουν **`null`** για το κενό σύνολο. Η συνάρτηση **`count`** επιστρέφει 0.

SQL: Ομαδοποίηση πλειάδων

- Ομαδοποίηση πλειάδων: Orders(orderno, month,cid,aid,pid,qty,amt)
  - Παρέχεται η δυνατότητα ομαδοποίησης των πλειάδων που αποτελούν την απάντηση σε μια ερώτηση σύμφωνα με τις κοινές τιμές κάποιων γνωρισμάτων.
  - Μπορούν επίσης να εφαρμοστούν συναρτήσεις συνάθροισης στις ομαδοποιημένες πλειάδες.
  - Παράδειγμα: η ερώτηση

```
select pid, sum(qty) from orders
group by pid;
```

θα επιστρέψει τα διακριτά pids μαζί με τη συνολική ποσότητα για την οποία έχουν γίνει παραγγελίες.

SQL: Ομαδοποίηση πλειάδων

Orders(orderno, month,cid,aid,pid,qty,amt)

- Όταν μια συνάρτηση συνάθροισης εμφανίζεται σε μια εντολή select η οποία περιέχει group-by, η συνάρτηση εφαρμόζεται σε όλες τις πλειάδες μιας ομάδας (δηλαδή όλες τις πλειάδες οι οποίες έχουν την ίδια τιμή στα γνωρίσματα για τα οποία γίνεται η ομαδοποίηση) και επιστρέφεται μια τιμή για κάθε ομάδα.
- Όλα τα γνωρίσματα τα οποία επιστρέφονται ως απάντηση πρέπει να έχουν μοναδική τιμή για κάθε συνδυασμό τιμών των γνωρισμάτων σύμφωνα με τα οποία γίνεται η ομαδοποίηση.
- Παράδειγμα: η ερώτηση

```
select pid, cid, sum(qty) from orders
 group by pid;
```

είναι λανθασμένη γιατί για ένα προϊόν δίνονται παραγγελίες από έναν ή περισσότερους πελάτες.

SQL: Ομαδοποίηση πλειάδων

Orders(orderno, month,cid,aid,pid,qty,amt)

- Η ομαδοποίηση μπορεί να γίνεται με περισσότερα από ένα γνωρίσματα
- Παράδειγμα: Υπολογίστε τη συνολική ποσότητα που παραγγέλνεται για κάθε προϊόν από κάθε πράκτορα.

```
select pid, aid, sum(qty) as TOTAL from orders
group by pid, aid;
```

pid	aid	TOTAL
p01	a01	3000
p01	a06	1800
p02	a02	400
p03	a03	1000
p03	a05	800

SQL: Ομαδοποίηση πλειάδων

Orders(orderno, month,cid,aid,pid,qty,amt)
Agents(aid, aname,city,percent)
Products(pid, pname,city, qty)

- Ομάδες πλειάδων μπορούν να σχηματιστούν με συνδυασμό σχέσεων
- Παράδειγμα: Βρείτε τα ονόματα και ids πρακτόρων, τα ονόματα και ids προϊόντων και τη συνολική ποσότητα που παραγγέλνει κάθε πράκτορας για τους πελάτες c02 και c03.

```
select aname, a.aid, pname, p.pid, sum(qty)
as TOTAL from orders o, products p, agents a
where o.pid=p.pid and o.aid=a.aid and o.cid
in ('c02','c03') group by a.aid, aname,
p.pid, pname;
```

aname	aid	pname	pid	TOTAL
Brown	a03	pencil	p05	2400
Brown	a03	razor	p03	1000
Black	a05	razor	p03	800

SQL: Ομαδοποίηση πλειάδων

- Υπολογισμός ερωτήσεων που περιέχουν group-by
  1. Υπολογίζεται το Καρτεσιανό γινόμενο των σχέσεων στο from
  2. Οι πλειάδες που δεν ικανοποιούν τις συνθήκες στο where αφαιρούνται
  3. Οι υπόλοιπες πλειάδες ομαδοποιούνται σύμφωνα με το group-by
  4. Υπολογίζονται οι εκφράσεις που επιστρέφονται ως απάντηση
  - Παράδειγμα: η ερώτηση

    ```
    select pid, sum(qty) from orders where
    sum(qty) >1000 group by pid;
    ```

    είναι λανθασμένη γιατί η συνθήκη στο where δε μπορεί να υπολογιστεί πριν γίνει η ομαδοποίηση των πλειάδων.

SQL: `having`

- Συνθήκες πάνω στις ομάδες των πλειάδων μπορούν να εκφραστούν με το having. Ο υπολογισμός αυτής της συνθήκης γίνεται μετά την ομαδοποίηση.
  - Παράδειγμα: Βρείτε τα ids των προϊόντων και τη συνολική ποσότητα που έχει παραγγελθεί, όταν η ποσότητα αυτή είναι μεγαλύτερη από 1000.

```
select pid, sum(qty) from orders
group by pid
having sum(qty)>1000;
```

- Αν δεν υπάρχει group-by αλλά υπάρχει having, τότε το σύνολο των πλειάδων θεωρείται ως μια ομάδα.

SQL: `having`

- Οι συνθήκες που εκφράζονται στο having μπορούν να περιλαμβάνουν μόνο γνωρίσματα τα οποία έχουν μοναδική τιμή για κάθε ομάδα.
  - Παράδειγμα: Βρείτε τα ids προϊόντων που έχουν παραγγελθεί από τουλάχιστον δύο πελάτες

```
select pid from orders
group by pid
having count(distinct cid)>=2;
```

  - Το γνώρισμα cid, ως έχει, δε μπορεί να συμμετέχει σε συνθήκη του having (δηλαδή η συνθήκη having cid=3; είναι λανθασμένη) γιατί για ένα προϊόν δίνονται παραγγελίες από έναν ή περισσότερους πελάτες.
  - Για τον ίδιο λόγο, το γνώρισμα cid δεν μπορεί να επιστρέφεται από την ερώτηση.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Αποτελούν μηχανισμό για τον έλεγχο της συνέπειας των δεδομένων.
- Χρησιμοποιούνται για να εξασφαλιστεί ότι μια βάση δεδομένων δεν θα βρεθεί σε ασυνεπή κατάσταση.
- Περιορισμοί ακεραιότητας στο μοντέλο Οντοτήτων – Σχέσεων:
  - ορισμός κλειδιών: η δήλωση ενός πρωτεύοντος (ή υποψήφιου κλειδιού) για ένα σύνολο οντοτήτων περιορίζει τις αποδεκτές εισαγωγές και ενημερώσεις σε αυτές ώστε να μην δημιουργούνται οντότητες με ίδια τιμή στο κλειδί.
  - περιορισμοί πληθικότητας: περιορίζουν το σύνολο των αποδεκτών σχέσεων μεταξύ συνόλων οντοτήτων
- Εν γένει, οι περιορισμοί ακεραιότητας μπορούν να είναι αυθαίρετα κατηγορήματα που αναφέρονται σε μια βάση δεδομένων.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Κατηγορήματα μεγάλης πολυπλοκότητας θα είναι δύσκολο να ελέγχονται.
- Περιοριζόμαστε σε τύπους περιορισμών οι οποίοι είναι εύκολο να ελεγχθούν μετά από κάθε αλλαγή που συμβαίνει στη βάση δεδομένων.
- Τύποι περιορισμών ακεραιότητας:
  1. Περιορισμοί Πεδίων Τιμών (Domain Constraints)
     - Στοιχειώδης τύπος περιορισμών ακεραιότητας
     - Χρησιμοποιείται για να ελεγχθεί αν τιμές γνωρισμάτων ανήκουν στα επιθυμητά πεδία τιμών

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Ορισμός περιορισμών πεδίων τιμών στην SQL:
  - Ορισμός πεδίου τιμών:
    e.g., `create domain person-name char(20);`
    ορίζει το `person-name` σαν ένα πεδίο τιμών αποτελούμενο από strings μήκους 20. Το `person-name` μπορεί ακόλουθα να χρησιμοποιηθεί σαν τύπος γνωρισμάτων.
  - Η κενή τιμή (`null`) ανήκει σε κάθε πεδίο τιμών.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Παραδείγματα:

```
create domain hourly-wage numeric(5,2)
constraint wage-value-test
check(value>=4.00);

create domain account-number char(10)
constraint account-number-null-test
check(value not null);

create domain account-type char(10)
constraint account-type-test check(value
in (“Checking”, “Savings”));
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

2. Περιορισμοί Αναφοράς (Referential Integrity Constraints)

- Χρησιμοποιούνται για να διασφαλιστεί ότι τιμές γνωρισμάτων που εμφανίζονται σε μια σχέση εμφανίζονται και σε άλλες σχέσεις (όταν αυτές έχουν κοινά γνωρίσματα)
- Εκκρεμείς Πλειάδες (dangling tuples): πλειάδες σε σχέσεις στις οποίες μπορεί να εφαρμοστεί join οι οποίες όμως δεν συμμετέχουν στο αποτέλεσμα του join.

  Δηλαδή, αν για την πλειάδα $t_r \in r$, δεν υπάρχει πλειάδα $t_s \in s$ έτσι ώστε $t_r [r \cap s] = t_s [r \cap s]$, τότε η πλειάδα t_r λέγεται εκκρεμής.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Παράδειγμα: έστω η σχέση account με σχήμα (account-number, branch-name, balance), η σχέση branch με σχήμα (branch-name, branch-city, assets). Έστω πλειάδα t στη σχέση account με t[branch name]="Atlantis". Αν στη σχέση branch δεν υπάρχει πλειάδα με αυτο το branch-name, τότε η πλειάδα t είναι εκκρεμής.
- Θέλουμε να έχουμε περιορισμούς οι οποίοι αποκλείουν τέτοιες καταστάσεις, ιδιαίτερα όταν το γνώρισμα στο οποίο εμφανίζονται «ανύπαρκτες» τιμές είναι ξένο κλειδί (foreign key).
- Ξένα κλειδιά: έστω r_1 και r_2 σχέσεις με κλειδιά K_1 και K_2 αντίστοιχα. Ένα υποσύνολο α του K_2 είναι ξένο κλειδί αναφερόμενο στο K_1, αν απαιτείται για κάθε πλειάδα t_2 στην r_2 να υπάρχει πλειάδα t_1 στην r_1 έτσι ώστε $t_1[K_1] = t_2[\alpha]$.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Απαιτήσεις αυτής της μορφής ονομάζονται περιορισμοί αναφοράς.
- Περιορισμοί αναφοράς στο μοντέλο Οντοτήτων-Σχέσεων
  - Αν ένα σχεσιακό σχήμα προκύπτει από την παραγωγή πινάκων απο διαγράμματα Οντοτήτων-Σχέσεων, τότε κάθε πίνακας που αντιστοιχεί σε μια σχέση στο διάγραμμα έχει περιορισμούς αναφοράς:

    αν η σχέση R είναι βαθμού N μεταξύ των οντοτήτων E_1, E_2, ..., E_n και K_i είναι το πρωτεύον κλειδί της E_i, τότε το σχήμα για την R θα είναι $K_1 \cup K_2 \cup ... \cup K_n$ και κάθε K_i είναι ξένο κλειδί.
  - Το σχήμα κάθε σχέσης (πίνακα) που αναπαριστά μια ασθενή οντότητα περιλαμβάνει ως ξένο κλειδί το πρωτεύον κλειδί της αντίστοιχης ισχυρής οντότητας.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Περιορισμοί αναφοράς και μεταβολή της ΒΔ.
  - Μεταβολές στην ΒΔ μπορούν να προκαλέσουν την παραβίαση περιορισμών αναφοράς.
  - Για κάθε είδος μεταβολής, υπάρχει μια συνθήκη η οποία πρέπει να ελεγχθεί για να διασφαλιστεί ότι δεν παραβιάζεται κανένας περιορισμός.
  - Έστω ότι ο περιορισμός αναφοράς περιγράφεται από την ακόλουθη συνθήκη:

$$\pi_{\alpha}\,(r_2) \subseteq \pi_{K}\,(r_1)$$

Περιορισμοί Ακεραιότητας (Integrity Constraints)

$$\pi_{\alpha}(r_2) \subseteq \pi_K(r_1)$$

a) Εισαγωγή: αν η πλειάδα t_2 εισαχθεί στη σχέση r_2, πρέπει να ελεγχθεί ότι υπάρχει πλειάδα t_1 στην r_1 έτσι ώστε $t_1[K] = t_2[\alpha]$. Άρα η συνθήκη που πρέπει να ελεγχθεί είναι: $t_2[\alpha] \in \pi_K(r_1)$

b) Διαγραφή: αν η πλειάδα t_1 διαγραφεί από την r_1 πρέπει να βρεθεί το σύνολο των πλειάδων της r_2 που αναφέρονται στην t_1: $\sigma_{\alpha = t1[K]}(r_2)$. Αν αυτό το σύνολο δεν είναι κενό, τότε είτε η διαγραφή δεν θα εκτελεστεί, είτε οι πλειάδες που αναφέρονται στην t_1 πρέπει να διαγραφούν επίσης.

   Η δεύτερη επιλογή μπορεί να οδηγήσει σε συνακόλουθες (cascaded) διαγραφές καθώς πλειάδες άλλων σχέσεων μπορεί να αναφέρονται σε πλειάδες που αναφέρονται στην t_1 κ.ο.κ.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

$$\pi_{\alpha}(r_2) \subseteq \pi_K(r_1)$$

c) Ενημέρωση: υπάρχουν δύο περιπτώσεις

I. Ενημέρωση στην r_2: αν μια πλειάδα t_2 της r_2 ενημερώνεται και η ενημέρωση μεταβάλλει την τιμή του ξένου κλειδιού α, τότε αν t_2' είναι η νέα πλειάδα, πρέπει να ελεγχθεί αν $t_2'[\alpha] \in \pi_K(r_1)$

II. Ενημέρωση στην r_1: αν μια πλειάδα t_1 της r_1 ενημερώνεται και η ενημέρωση μεταβάλλει την τιμή του πρωτεύοντος κλειδιού K, τότε πρέπει να βρεθεί το σύνολο $\sigma_{\alpha= t1[K]}(r_2)$. Αν το σύνολο αυτό είναι μη-κενό, τότε είτε η ενημέρωση δεν εκτελείται είτε η ενημέρωση γίνεται με συνακόλουθες ενημερώσεις (όπως στην περίπτωση της διαγραφής).

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Ορισμός περιορισμών αναφοράς στην SQL:

```
create table customer (customer-name
char(20) not null, customer-street
char(30), customer-city char(30),
primary key (customer-name));

create table branch (branch-name char(15)
not null, branch-city char(30), assets
integer, primary key (branch-name),
     check (assets>=0));
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

```
create table account (account-number
char(10) not null, branch-name char(15),
balance integer,  primary key (account-
number), foreign key (branch-name)
references branch, check(balance >=0));

create table depositor (customer-name
char(20) not null, account-number char(10)
not null,    primary key (customer-name,
account-number), foreign key (customer-name
references customer,   foreign key
(account-number) references account);
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Καθορίζεται επίσης το πως αντιμετωπίζεται μια παραβίαση περιορισμού:

```
create table account (account-number
char(10) not null, branch-name char(15),
balance integer,    primary key
(account-number),
foreign key (branch-name) references
branch,    on delete cascade,
           on update cascade,
check(balance >=0));
```

Διαγραφή μιας πλειάδας από τη σχέση branch ακολουθείται από τη διαγραφή των πλειάδων της σχέσης account που την αναφέρουν. Παρόμοια, στην περίπτωση της ενημέρωσης, ενημερώνονται τα αντίστοιχα γνωρίσματα της σχέσης account.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Άλλες ενέργειες για την αντιμετώπιση της παραβίασης περιορισμών περιλαμβάνουν την εισαγωγή κενής τιμής στο πεδίο το οποίο αναφέρεται σε πεδίο πλειάδας άλλης σχέσης η οποία διαγράφεται ή ενημερώνεται.
- Αν η παραβίαση ενός περιορισμού δεν μπορεί να διορθωθεί με διαγραφές / ενημερώσεις , τότε η πράξη που προκαλεί την παραβίαση διακόπτεται και αναιρούνται οι πράξεις που εκτελέστηκαν.
- Χειρισμός κενών τιμών:
  - Όλα τα γνωρίσματα του πρωτεύοντος κλειδιού θεωρούνται `not null`
  - Γνωρίσματα ξένων κλειδιών μπορούν να δέχονται κενές τιμές. Οι κενές τιμές δεν παραβιάζουν τους περιορισμούς αναφοράς.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Δηλώσεις (assertions)
  - Κατηγορήματα τα οποία εκφράζουν συνθήκες οι οποίες πρέπει να ικανοποιούνται σε κάθε στιγμιότυπο μιας ΒΔ.
  - Οι περιορισμοί πεδίου και αναφοράς είναι ειδικές περιπτώσεις τέτοιων κατηγορημάτων.
  - Η SQL παρέχει την εντολή `create assertion.`
  - Παράδειγμα: the sum of all loan amounts for each branch must be less than the sum of all account balances at the branch

```
create assertion sum-constraint
check (not exists (select * from branch
where       (select sum(amount) from loan
where loan.branch-name = branch.branch-name)
>= (select sum(balance) from account where
account.branch-name = branch.branch-name)));
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Δηλώσεις (assertions)
  - Όταν δημιουργείται μια δήλωση, ελέγχεται αν ισχύει. Αν ναι, τότε μεταβολές στη βάση δεδομένων επιτρέπονται μόνο αν δεν παραβιάζουν τη δήλωση.
  - Ο έλεγχος δηλώσεων έχει μεγάλο κόστος. Τα ΣΔΒΔ συνήθως δεν βελτιστοποιούν τον έλεγχο δηλώσεων.
- Ενεργοί κανόνες (active rules, triggers)
  - Ενεργοί κανόνες εκτελούνται αυτόματα σαν αποτέλεσμα μιας μεταβολής στη βάση δεδομένων.
  - Ο ορισμός των κανόνων περιλαμβάνει τον προσδιορισμό των συνθηκών κάτω από τις οποίες ο κανόνας θα εκτελείται και τις ενέργειες που θα εκτελεστούν.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Παράδειγμα: αν κάποιος αποσύρει περισσότερα χρήματα από όσα έχει στο λογαριασμό του, αυτόματα δημιουγείται ένα δάνειο με το ποσό που υπερβαίνει το υπόλοιπο του λογαριασμού του.

```
define trigger overdraft on update of account T
(if new T.balance < 0 then
(insert into loan values
   (T.branch-name, T.account-no, -new T.balance)
(insert into borrower
   (select customer-name, account-no from depositor
   where T.account-no = depositor.account-no)
update account S
set S.balance=0 where S.account-no=T.account-no));
```

- Η λέξη **new** συμβολίζει την ενημερωμένη πλειάδα.