SQL – Structured Query Language

- Κατηγορήματα Σύγκρισης με Ποσοδείκτες (Quantified Comparison Predicates)
  - χρησιμοποιούνται για τη σύγκριση της τιμής μιας έκφρασης με το αποτέλεσμα μιας εντολής select
  - γενική μορφή: ***expr θ* {any | all} (*subselect*),**
    θ ∈ {<, ≤, =, ≠, >, ≥}
  - συνολικά 12 κατηγορήματα μπορούν να οριστούν
  - Η έκφραση ***expr θ* any (*subselect*)** είναι αληθής αν για τουλάχιστον ένα στοιχείο s στο αποτέλεσμα του ***subselect*** η έκφραση ***expr θ s*** είναι αληθής
  - Η έκφραση ***expr θ* all (*subselect*)** είναι αληθής αν για όλα τα στοιχεία s στο αποτέλεσμα του ***subselect*** η έκφραση ***expr θ s*** είναι αληθής

SQL – Data Manipulation Language

- Παραδείγματα:
  1. Find the ids of agents with minimum percent commission

```
select aid from agents
where percent <= all
                (select percent from agents);
```

  2. Find the names of all customers that have the same discount as any of the customers in Dallas or Boston

```
select cname from customers
where discnt = any
      (select discnt from customers
       where city ='Dallas' or city = 'Boston');
```

SQL – Data Manipulation Language

3. Find the ids of customers with discount smaller than that of any customer who lives in Dallas.

```
select cid from customers
where discnt < any
        (select discnt from customers
         where city = 'Dallas');
```

- Η έκφραση αυτή είναι λάθος! Ορθή έκφραση:

```
select cid from customers
where discnt < all
        (select discnt from customers
         where city = 'Dallas');
```

SQL – Data Manipulation Language

- Παρατηρήσεις:
  - Το κατηγόρημα *expr* = **any** (*subselect*) έχει την ίδια σημασιολογία με την έκφραση *expr* **in** (*subselect*)
  - Η έκφραση *expr* **not in** (*subselect*) δεν είναι ισοδύναμη με το κατηγόρημα *expr* ≠ **any** (*subselect*)
  - Είναι ισοδύναμη με το κατηγόρημα *expr* ≠ **all** (*subselect*)
- Το κατηγόρημα **exists**
  - Χρησιμοποιείται για να ελεγχθεί αν το αποτέλεσμα ενός subselect είναι κενό.
  - Γενική μορφή: [**not**] **exists** (*subselect*)
  - **exists** (*subselect*) είναι αληθής αν το αποτέλεσμα του subselect είναι μη-κενό σύνολο

SQL – Data Manipulation Language

- Παραδείγματα:

1. Find the names of the customers that place an order through agent a05.

```
select cname from customers, orders
where customers.cid=orders.cid and aid='a05';
```

or

```
select cname from customers
where exists (select * from orders
where customers.cid=orders.cid and aid='a05');
```

Στο Σ.Λ. Πλειάδων: $\{t^{(1)} \mid \exists u^{(4)} (customers(u) \wedge (u[2]=t[1]) \wedge \exists v^{(7)} (orders(v) \wedge (v[3]=u[1]) \wedge (v[4]='a05')))\}$

SQL – Data Manipulation Language

2. Find the ids of customers that order both products p01 and p07.

```
select cid from orders x
where x.pid = 'p01' and exists
(select * from orders where
 cid=x.cid and pid = 'p07');
```

3. Find the names of customers that do not place an order through agent a05.

```
select cname from customers
where not exists (select * from orders where
orders.cid = customers.cid and aid = 'a05');
```

SQL – Data Manipulation Language

- Ποιο είναι το αποτέλεσμα της εντολής

```
select cname from customers, orders where
not (orders.cid = customers.cid and aid='a05');
```

σε σχέση με την προηγούμενη ερώτηση;

  - επιστρέφει τους πελάτες ο οποίοι δίνουν παραγγελίες με πράκτορες διαφορετικούς από τον a05 αλλά οι οποίοι ενδέχεται να δίνουν παραγγελίες και μέσω του a05.

- Η ερώτηση μπορεί να εκφραστεί και ως:

```
select cname from customers where cid not in
(select cid from orders where aid='a05');
```
ή
```
select cname from customers where cid ≠ all
(select cid from orders where aid = 'a05');
```

SQL – Data Manipulation Language

4. Το κατηγόρημα **not exists** χρησιμοποιείται για την έκφραση του τελεστή της διαφοράς :

   - αν R και S είναι συμβατές σχέσεις με σχήμα $\{A_1, \ldots, A_n\}$, η διαφορά R – S μπορεί να εκφραστεί ως:

```
select A1, …, An from R where not exists
(select * from S where S.A1 = R.A1 and S.A2 =
R.A2 and … and S.An = R.An);
```

SQL – Data Manipulation Language

5. Find the names of the cities that have customers that order product p01

```
select distinct city from customers where cid
in (select cid from orders where pid='p01');

select distinct city from customers where
cid = any (select cid from orders where
pid='p01');

select distinct city from customers c where
exists (select * from orders where cid=c.cid
and pid='p01');
```

SQL – Data Manipulation Language

```
select distinct city from customers, orders
where orders.cid=customers.cid and
orders.pid = 'p01';

select distinct city from customers where
'p01' in (select pid from orders where cid =
customers. cid);
```

- Όλες αυτές οι μορφές είναι ισοδύναμες.
- Η γλώσσα θα είχε την ίδια εκφραστική δύναμη αν το κατηγόρημα **in** και τα κατηγορήματα σύγκρισης με ποσοδείκτες παραλείπονταν.
- Δεν μπορεί όμως να παραληφθεί το κατηγόρημα **exists**.

SQL – Data Manipulation Language

- Τελεστής **union**
  - συνδυάζει subselects τα οποία παράγουν συμβατές σχέσεις
  - γενική μορφή: *subselect* {**union** [**all**] *subselect*}
  - περιορισμός: τα subselects δεν μπορούν να περιέχουν τον τελεστή **union**
  - Παράδειγμα: find the cities where either a customer or an agent is based

```
select city from customers union select city
from agents;
```

ή για να επιτρέψομε επαναλαμβανόμενες πλειάδες

```
select city from customers union all
select city from agents;
```

SQL: Διαίρεση

- Διαίρεση: η SQL δεν παρέχει τελεστή για διαίρεση
- Παράδειγμα: Find the ids of customers that **place** orders through **all** agents based in New York
  - Ή ισοδύναμα: find the ids of customers that are such that, **there does not exist** an agent based in New York that **does not place** an order for each of these customers.

```
select cid from customers where not exists
(select * from agents where city="New York"
and not exists
        (select * from orders where
        orders.cid=customers.cid and
        orders.aid=agents.aid));
```

SQL: Διαίρεση

- Παράδειγμα: Find the ids of agents based in New York or Dallas who **place** an order for **all** products costing more than 1$.
  - Ή ισοδύναμα: find the ids of agents that are based in New York or Dallas and are such that, **there does not exist** a product costing more than 1$ that they **do not** order.

```
select aid from agents where (city="New York"
or city = "Dallas") and not exists
(select pid from products where price > 1.00
and not exists
        (select * from orders where
        orders.pid=products.pid and
        orders.aid=agents.aid));
```

SQL: Διαίρεση

- Παράδειγμα: Find the ids of products that **are ordered** by **all** customers in Dallas.
  - Ή ισοδύναμα: find the ids of products that are such that, **there does not exist** a customer in Dallas that **does not order** these products.

```
select pid from products where not exists
(select cid from customers where city="Dallas"
and not exists
        (select * from orders where
        orders.pid=products.pid and
        orders.cid=customers.cid));
```

- Η έκφραση της διαίρεσης με αυτό τον τρόπο βασίζεται στην ισοδυναμία:

$$\forall z\ \exists y\ p(z,y) \equiv \neg\exists z\ \neg\exists y\ p(z,y)$$

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

- Παραδείγματα:
  1. Determine the total amount of all orders

     ```
     select sum(dollars) from orders;
     ```
  2. Determine the total quantity of product p03 that has been ordered.

     ```
     select sum(quantity) as TOTAL from orders
     where pid='p03';
     ```
  3. Find the total number of customers.

     ```
     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

4. Find the total number of cities in which customers are based.

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

5. Find the ids of customers whose discount is less than the maximum discount.

   Η έκφραση:

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

   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.

CUSTOMERS

cid	cname	city	discnt

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

Η τιμή στο γνώρισμα discnt θα είναι null.

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: Find the average discount of customers

    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 :
  - 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

- 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

- Η ομαδοποίηση μπορεί να γίνεται με περισσότερα από ένα γνωρίσματα
- Παράδειγμα: Calculate the total product quantity ordered of each product by each individual agent

```
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

- Ομάδες πλειάδων μπορούν να σχηματιστούν με συνδυασμό σχέσεων
- Παράδειγμα: Find the names and ids of agents, the names and ids of products and the total quantity that each agent supplies the product to customers c02 and 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: Find all ids of products and the total quantity ordered when this quantity exceeds 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: Find all ids of products that have been ordered by at least two customers

```
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.