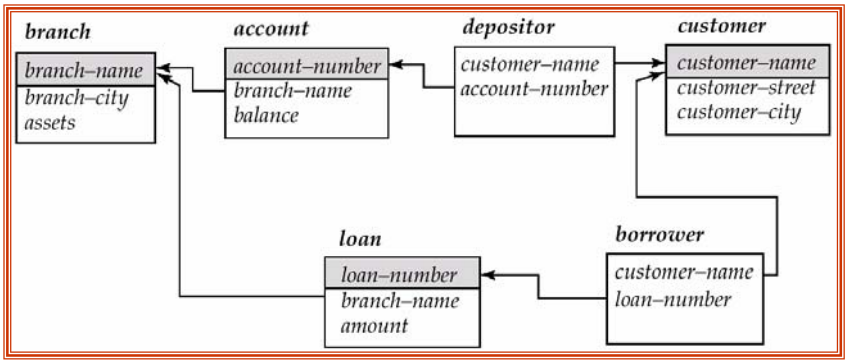
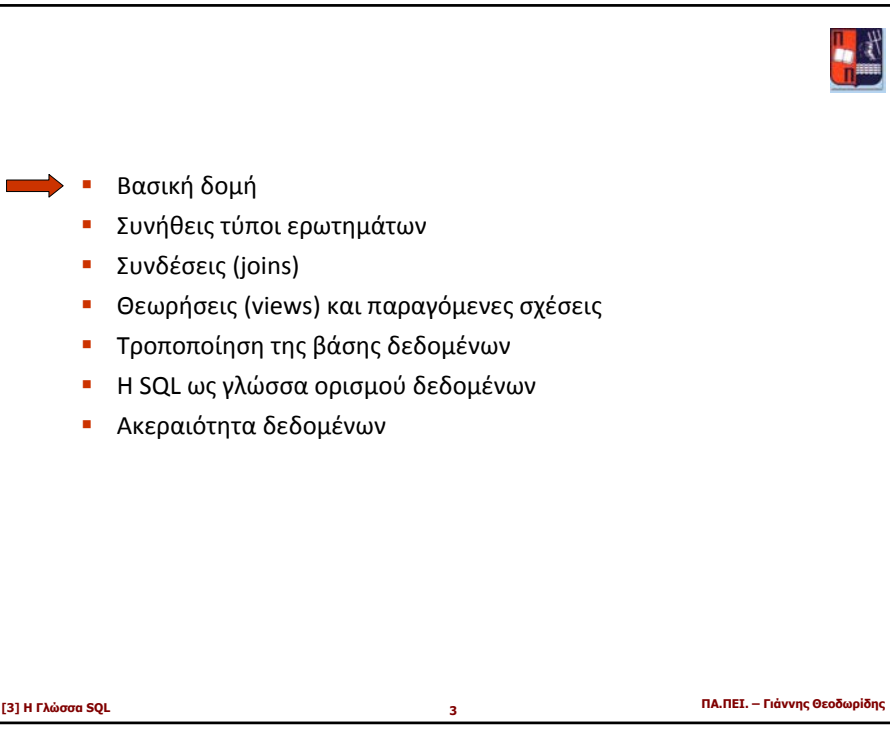
Η Γλώσσα SQL

- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Βασική πηγή διαφανειών: Silberschatz et al., "Database System Concepts", 4/e
Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς (http://infolab.cs.unipi.gr/)
έκδοση: Φεβ. 2010

Παράδειγμα ΒΔ (Τράπεζα)





- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Βασική Δομή

- Η SQL βασίζεται στις πράξεις μεταξύ συνόλων και σχέσεων.
- Μια χαρακτηριστική ερώτηση σε SQL έχει την παρακάτω μορφή:

  select $A_1, A_2, \ldots, A_n$
  from $r_1, r_2, \ldots, r_m$
  where P

  - Τα A_i αναπαριστούν χαρακτηριστικά
  - Τα r_i αναπαριστούν σχέσεις
  - Το P είναι ένα κατηγόρημα (ουσιαστικά μια συνθήκη).
- Το αποτέλεσμα μιας SQL ερώτησης είναι μια σχέση.

- Το ερώτημα αυτό είναι ισοδύναμο με την ακόλουθη έκφραση ΣΑ:

$$\Pi_{A1, A2, \ldots, An}(\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$$

Το τμήμα select (1)

- Το τμήμα **select** χρησιμοποιείται για την προβολή των χαρακτηριστικών που θέλουμε να υπάρχουν στο αποτέλεσμα της ερώτησης.
- Παράδειγμα: «Βρες τα ονόματα των υποκαταστημάτων που έχουν συνάψει δάνεια»

  select *branch-name*
  from *loan*
- Ο αστερίσκος στο τμήμα select δηλώνει "όλα τα χαρακτηριστικά". Για παράδειγμα:

  select * **from** *loan*

- Η ισοδύναμη έκφραση ΣΑ είναι η εξής:

  $\Pi_{\text{branch-name}}(loan)$

Το τμήμα select (2)

- Η SQL επιτρέπει πολλαπλές εμφανίσεις της ίδιας πλειάδας σε μια σχέση, όπως και στο αποτέλεσμα μιας ερώτησης. Για την απαλοιφή των πολλαπλών εμφανίσεων, εισάγουμε τη δεσμευμένη λέξη **distinct** μετά το **select.**

  Παράδειγμα: «Βρες τα (διακριτά) ονόματα των υποκαταστημάτων που έχουν συνάψει δάνεια»

  select distinct *branch-name*

  from *loan*
- Το τμήμα **select** μπορεί να περιέχει αριθμητικές εκφράσεις περιλαμβάνοντας τις πράξεις +, –, ∗ και / ανάμεσα σε σταθερές ή χαρακτηριστικά πλειάδων.
- Το ερώτημα:

  select *loan-number, branch-name, amount* ∗ 100
  from *loan*

  θα επιστρέψει μια σχέση η οποία είναι η ίδια με τη σχέση *loan*, εκτός από το χαρακτηριστικό *amount*, το οποίο έχει πολλαπλασιαστεί επί 100.

Το τμήμα where

- Το τμήμα **where** αποτελείται από ένα κατηγόρημα που περιέχει χαρακτηριστικά των σχέσεων που εμφανίζονται στην πρόταση **from**.
- Παράδειγμα: «Βρες τα δάνεια που έγιναν στο υποκατάστημα Perryridge με ποσά άνω των 1200»

  select * **from** *loan*
  where *branch-name* = 'Perryridge' **and** *amount* > 1200

- Τα αποτελέσματα των συγκρίσεων μπορούν να συνδυαστούν με χρήση των λογικών τελεστών **and, or** και **not.**
- Οι συγκρίσεις μπορούν να εφαρμοστούν σε αποτελέσματα αριθμητικών εκφράσεων.

- Η ισοδύναμη έκφραση ΣΑ είναι η εξής:

  σ *branch-name* = 'Perryridge' **and** *amount* > 1200 (*loan*)

Το τμήμα from

- Το τμήμα **from** αντιστοιχεί στο Καρτεσιανό γινόμενο μεταξύ σχέσεων. Παραθέτει τις σχέσεις που πρέπει να σαρωθούν για την αποτίμηση της έκφρασης.
- Παράδειγμα: Καρτεσιανό γινόμενο *borrower x loan*

  select *

  from *borrower, loan*

- Παράδειγμα: «Βρες το όνομα, αριθμό δανείου και ποσό δανείου για όλους τους πελάτες, οι οποίοι έχουν πάρει δάνειο από το υποκατάστημα Perryridge»

  select *customer-name, borrower.loan-number, amount*
  from *borrower, loan*
  where *borrower.loan-number = loan.loan-number* **and**
  branch-name = 'Perryridge'

- Η ισοδύναμη έκφραση ΣΑ είναι η εξής:

  borrower x loan

Η λειτουργία Rename

- Η SQL επιτρέπει τη μετονομασία σχέσεων και χαρακτηριστικών με χρήση της πρότασης **as**:

  old-name **as** *new-name*

- Παράδειγμα: «Βρες το όνομα, αριθμό δανείου και ποσό δανείου για όλους τους πελάτες (η στήλη *loan-number* να μετονομαστεί σε *loan-id*)»

  select *customer-name, borrower.loan-number* **as** *loan-id, amount*
  from *borrower, loan*
  where *borrower.loan-number = loan.loan-number*

Μεταβλητές πλειάδων

- Οι μεταβλητές πλειάδων ορίζονται στην πρόταση **from** με χρήση της πρότασης **as**.
- Παράδειγμα: «Βρες τα ονόματα των πελατών και τους αντίστοιχους αριθμούς δανείων για όλους τους πελάτες που έχουν πάρει δάνειο από κάποιο υποκατάστημα»

  select *customer-name, b.loan-number, l.amount*
  from *borrower* **as** *b, loan* **as** *l*
  where *b.loan-number = l.loan-number*

- Παράδειγμα: «Βρες τα υποκαταστήματα που έχουν αποθεματικό μεγαλύτερο του αποθεματικού ενός (οποιουδήποτε) υποκαταστήματος του Brooklyn»

  select distinct *b1.branch-name*
  from *branch* **as** *b1, branch* **as** *b2*
  where *b1.assets > b2.assets* **and** *b2.branch-city* = 'Brooklyn'

Πράξεις με συμβολοσειρές

- Η SQL παρέχει έναν τελεστή ταιριάσματος συμβολοσειρών (τον τελεστή **like**) για συγκρίσεις με χρήση δύο ειδικών χαρακτήρων '**%**' και '**_**':
  - (%). Ο χαρακτήρας **%** ταιριάζει οποιαδήποτε υπο-συμβολοσειρά.
  - (_). Ο χαρακτήρας **_** ταιριάζει οποιοδήποτε χαρακτήρα.
- Παράδειγμα: «Βρες τα ονόματα των πελατών, των οποίων ο δρόμος της κατοικίας περιλαμβάνει τη λέξη "Main".

```
select customer-name
from customer
where customer-street like '%Main%'
```

- Η SQL υποστηρίζει ένα πλήθος από πράξεις σε συμβολοσειρές όπως:
  - αλληλουχία (με χρήση του "||"),
  - μετατροπή από κεφαλαία σε πεζά (και αντίστροφα),
  - εύρεση του μήκους της συμβολοσειράς, κλπ.

Διάταξη των πλειάδων

- Παράδειγμα: « Βρες τα ονόματα των πελατών που έχουν πάρει δάνειο από το υποκατάστημα Perryridge και εμφάνισέ τα σε αλφαβητική σειρά»

```
select distinct customer-name
from   borrower, loan
where borrower loan-number = loan.loan-number and
        branch-name = 'Perryridge'
order by customer-name
```

- Μπορούμε να καθορίσουμε **desc** για φθίνουσα σειρά ή **asc** για αύξουσα σειρά για κάθε χαρακτηριστικό. Προκαθορισμένη σειρά είναι η αύξουσα.
  - Παράδειγμα: ... **order by** *customer-name* **desc**

- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Πράξεις συνόλων (1)

- Οι πράξεις συνόλων **union**, **intersect** και **except** εφαρμόζονται σε σχέσεις
- Καθεμιά από τις παραπάνω πράξεις αυτομάτως αφαιρεί τις διπλοεγγραφές. Για να διατηρηθούν οι πολλαπλές εμφανίσεις των εγγραφών χρησιμοποιούνται οι παρακάτω εκδοχές **union all**, **intersect all** και **except all**.
- Υποθέστε ότι μια πλειάδα εμφανίζεται *m* φορές στην *r* και *n* φορές στην *s*, τότε, εμφανίζεται:
  - *m + n* φορές στην *r* **union all** *s*
  - min(*m,n)* φορές στην *r* **intersect all** *s*
  - max(0, *m – n)* φορές στην *r* **except all** *s*
- Αντιστοιχούν στις πράξεις "$\cup$", "$\cap$", "$-$" της ΣΑ .

Πράξεις συνόλων (2)

- Παράδειγμα: «Βρες τους πελάτες που έχουν πάρει δάνειο ή διαθέτουν λογαριασμό»

  (**select** *customer-name* **from** *depositor*)
  union (**select** *customer-name* **from** *borrower)*

- Παράδειγμα: «Βρες τους πελάτες που έχουν πάρει δάνειο και διαθέτουν λογαριασμό»

  (**select** *customer-name* **from** *depositor*)
  intersect (**select** *customer-name* **from** *borrower)*

- Παράδειγμα: «Βρες τους πελάτες που διαθέτουν λογαριασμό και δεν έχουν πάρει δάνειο»

  (**select** *customer-name* **from** *depositor*)
  except (**select** *customer-name* **from** *borrower)*

Συναθροιστικές συναρτήσεις (1)

- Αυτές οι συναρτήσεις εφαρμόζονται στις τιμές μιας στήλης μιας σχέσης και επιστρέφουν μια τιμή.

  avg / min / max: μέση / ελάχιστη / μέγιστη τιμή

  sum / count: άθροισμα / πλήθος τιμών

- Παράδειγμα: «Βρες το μέσο υπόλοιπο των λογαριασμών στο υποκατάστημα Perryridge»

  select avg *(balance)* **from** *account*
  where *branch-name* = 'Perryridge'

- Παράδειγμα: «Βρες το πλήθος των πελατών της τράπεζας»

  select count (*) **from** *customer*

- Παράδειγμα: «Βρες το πλήθος των καταθετών της τράπεζας»

  select count (distinct *customer-name)*
  from *depositor*

Συναθροιστικές συναρτήσεις (2)

- Ομαδοποίηση εγγραφών – το τμήμα **group by**
- Παράδειγμα: «Βρες το πλήθος των καταθετών σε κάθε υποκατάστημα της τράπεζας».

```
select branch-name, count (distinct customer-name)
from depositor as d, account as a
where d.account-number = a.account-number
group by branch-name
```

- Σημείωση: τα χαρακτηριστικά που εμφανίζονται στο τμήμα **select** πρέπει να είναι είτε μεταξύ αυτών στα οποία εφαρμόζονται οι συναθροιστικές συναρτήσεις, είτε μεταξύ αυτών που εμφανίζονται στη λίστα **group by**.

Συναθροιστικές συναρτήσεις (3)

- Συναθροιστικές συναρτήσεις πάνω στις ομάδες εγγραφών – το τμήμα **having**
- Παράδειγμα: «Βρες τα υποκαταστήματα που έχουν μέσο υπόλοιπο λογαριασμών άνω των 2.000»

```
select branch-name, avg (balance)
from account
group by branch-name
having avg (balance) > 2000
```

- Σημείωση: Τα κατηγορήματα (συνθήκες) στο τμήμα **having** εφαρμόζονται μετά το σχηματισμό των ομάδων, ενώ τα κατηγορήματα στο τμήμα **where** εφαρμόζονται πριν το σχηματισμό των ομάδων.

Η τιμή null

- Είναι πιθανό κάποιες πλειάδες για κάποια από τα χαρακτηριστικά τους να έχουν μια κενή τιμή, η οποία δηλώνεται με **null**.
- Το **null** δηλώνει μια άγνωστη τιμή ή μια τιμή που δεν υπάρχει.
- Το κατηγόρημα **is null** (και όχι = null) μπορεί να χρησιμοποιηθεί για τον έλεγχο τιμών null.
  - Π.χ. «Βρες τα δάνεια που δεν έχουν καταχωρημένο το ποσό δανείου»

```
select loan-number
from loan
where amount is null
```


Οι τιμές null και οι συναθροίσεις

- Το αποτέλεσμα μιας αριθμητικής έκφρασης που περιέχει το **null** είναι **null**.
  - Π.χ. 5 + null επιστρέφει null
- Παρ' όλα αυτά, οι συναθροιστικές συναρτήσεις απλώς αγνοούν τα nulls.
- Παράδειγμα: «Βρες το συνολικό ύψος των δανείων»

```
select sum (amount)
from loan
```

  - Το παραπάνω ερώτημα SQL αγνοεί τις τιμές null
  - Το αποτέλεσμα θα είναι null μόνο εάν η στήλη amount είναι γεμάτη από τιμές null

Εμφωλιασμένα υποερωτήματα (1)

- Η SQL παρέχει ένα μηχανισμό για το **εμφώλιασμα υποερωτημάτων** (**subquery nesting**).
- Ένα υποερώτημα είναι μια **select-from-where** έκφραση, η οποία είναι εμφωλιασμένη σε μια άλλη ερώτηση.
- Μια κοινή χρήση των υποερωτημάτων είναι η πραγματοποίηση ελέγχων για συμμετοχή σε σύνολα, για συγκρίσεις συνόλων και για την πληθικότητα συνόλων.

Εμφωλιασμένα υποερωτήματα (2)

Παραδείγματα

- «Βρες τους πελάτες που έχουν λογαριασμό στην τράπεζα και έχουν πάρει δάνειο από αυτήν»

```
select distinct customer-name from borrower
where customer-name in
        (select customer-name from depositor)
```

- «Βρες τους πελάτες που έχουν πάρει δάνειο από την τράπεζα, αλλά δεν έχουν λογαριασμό σε αυτή»

```
select distinct customer-name from borrower
where customer-name not in
        (select customer-name from depositor)
```

Σύγκριση συνόλων

- Παράδειγμα: «Βρες τα υποκαταστήματα που έχουν αποθεματικό μεγαλύτερο του αποθεματικού ενός (οποιουδήποτε) υποκαταστήματος του Brooklyn»

  select distinct *b1.branch-name*
  from *branch* **as** *b1, branch* **as** *b2*
  where *b1.assets* > *b2.assets* **and** *b2.branch-city* = 'Brooklyn'

- Το ίδιο ερώτημα με χρήση της πρότασης **> some**

  select *branch-name*
  from *branch*
  where *assets* > **some**
  (select *assets* **from** *branch*
  where *branch-city* = 'Brooklyn')

Ορισμός της πρότασης some

- F ? **some** *r* ⇔ ∃ *t* ∈ *r* τέτοιο ώστε *(*F ? *t*)
  όπου ? είναι ένας από τους τελεστές σύγκρισης: <, ≤, >, =, ≠

(5< **some** {0, 5, 6}) = true (διαβάζεται: 5 < κάποια πλειάδα στη σχέση)

(5< **some** {0, 5}) = false (5 = **some** {0, 5}) = true

(5 ≠ **some** {0, 5}) = true (αφού 0 ≠ 5)

(= **some**) ≡ **in**
Παρ΄ όλα αυτά, (≠ **some**) ≢ **not in**

Ορισμός της πρότασης all

- F ? **all** *r* ⇔ ∀ *t* ∈ *r* ισχύει (F ? *t)*

(5< **all** {0, 5, 6}) = false

(2< **all** {4, 5}) = true

(5 = **all** {0, 5}) = false

(5 ≠ **all** {4, 6}) = true (αφού 5 ≠ 4 και 5 ≠ 6)

(≠ **all**) ≡ **not in**
Παρ' όλα αυτά, (= **all**) ≠ **in**

Παράδειγμα

- «Βρες τα υποκαταστήματα που έχουν αποθεματικό μεγαλύτερο του αποθεματικού όλων των υποκαταστημάτων του Brooklyn»

```
select branch-name
from branch
where assets > all
    (select assets from branch
    where branch-city = 'Brooklyn')
```

Έλεγχος για κενές σχέσεις

- Ο τελεστής **exists** επιστρέφει τιμή **true** εάν το υποερώτημα που παίρνει ως όρισμα είναι μη κενό.

  exists $r \Leftrightarrow r \neq \emptyset$ **not exists** $r \Leftrightarrow r = \emptyset$

- Σημειώστε ότι $X - Y = \emptyset \Leftrightarrow X \subseteq Y$
- Παράδειγμα: «Βρες τους πελάτες που έχουν λογαριασμούς σε όλα τα υποκαταστήματα που βρίσκονται στο Brooklyn»

```
select distinct S.customer-name from depositor as S
where not exists (
      (select branch-name from branch
            where branch-city = 'Brooklyn')
except
      (select R.branch-name from depositor as T, account as R
            where T.account-number = R.account-number and
                  S.customer-name = T.customer-name))
```


Ερωτήσεις τύπου top-N (1)

- Κάποιες φορές θέλουμε να προσδιορίσουμε πόσες πλειάδες θα βγουν στο αποτέλεσμα.
- Δύσκολο στην SQL !!
- Παράδειγμα: Βρείτε τους λογαριασμούς με τα 10 μεγαλύτερα υπόλοιπα.

```
select account-number, balance
from account as A
where 10 >      (select count(*)
                 from account as B
                 where A.balance < B.balance)
and balance is not null
order by balance desc
```

- για το λόγο αυτό, τα περισσότερα συστήματα έχουν ειδική υποστήριξη αυτής της λειτουργίας (οπότε προκύπτουν ζητήματα μεταφερσιμότητας).

Ερωτήσεις τύπου top-N (2)

- MySQL:

```
select account-number, balance
from account
where balance is not null
order by balance desc
limit 10
```

- Oracle:

```
select account-number, balance
from (
    select account-number, balance
    from account
    where balance is not null
    order by balance desc)
where rownum <= 10
```

- SQL Server:

```
select top 10 account-number, balance
from account
where balance is not null
order by balance desc
```

(υπογραμμισμένα είναι τα ειδικά keywords που χρησιμοποιεί το κάθε σύστημα)

- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Σύνδεση σχέσεων (join)

- Οι λειτουργίες σύνδεσης (join) παίρνουν ως είσοδο δυο σχέσεις και επιστρέφουν ως αποτέλεσμα μια άλλη σχέση.
- Αυτές οι επιπρόσθετες πράξεις χρησιμοποιούνται συνήθως ως εκφράσεις υποερωτημάτων στο τμήμα **from**.
- **Συνθήκη Join**: καθορίζει ποιες πλειάδες στις δυο σχέσεις ταιριάζουν και ποια χαρακτηριστικά εμφανίζονται στο αποτέλεσμα του join.
- **Τύπος Join**: καθορίζει πώς θα χειριστούμε τις πλειάδες της μίας σχέσης, οι οποίες δεν ταιριάζουν (βάσει της συνθήκης του join) με κάποια πλειάδα της άλλης σχέσης.

Τύποι Join
inner join
left outer join
right outer join
full outer join

Συνθήκες Join
natural
on <predicate>
using (A_1, A_2, ..., A_n)

Τύποι join

$A \bowtie_c B$ σχέσεις A, B, συνθήκη c

- **Inner Join**: το αποτέλεσμα περιλαμβάνει **μόνο** τις πλειάδες των A και B που ταιριάζουν μεταξύ τους (σύμφωνα με την συνθήκη c)
- **Outer Join**: το αποτέλεσμα περιλαμβάνει **επιπλέον** και πλειάδες (της A ή/και της B) που δεν ταιριάζουν, ως εξής:
  - **Left outer join**: περιλαμβάνονται οι πλειάδες της A που δεν αντιστοιχούν σε πλειάδες της B
  - **Right outer join**: περιλαμβάνονται οι πλειάδες της B που δεν αντιστοιχούν σε πλειάδες της A
  - **Full outer join**: περιλαμβάνονται τόσο οι πλειάδες της A που δεν αντιστοιχούν σε πλειάδες της B όσο και οι πλειάδες της B που δεν αντιστοιχούν σε πλειάδες της A
  - Για κάθε πλειάδα της μιας σχέσης που δεν έχει αντίστοιχη πλειάδα στην άλλη σχέση, τα χαρακτηριστικά που δεν βρίσκουν αντιστοιχία παίρνουν τιμή **null**.

Συνθήκες join

- **Natural**

  ελέγχει τα χαρακτηριστικά των σχέσεων A και B με το ίδιο όνομα. Το αποτέλεσμα περιλαμβάνει πλειάδες των A, B που έχουν ίδια τιμή πάνω σε αυτά τα χαρακτηριστικά.
- **On <predicate>**

  δηλώνεται ένα κατηγόρημα (ουσιαστικά μια συνθήκη) που υπαγορεύει τον τρόπο που θα ταιριάξουν οι πλειάδες της A με τις πλειάδες της B.
- **Using (A_1, A_2, …, A_n)**

  περιλαμβάνει μια λίστα με χαρακτηριστικά (που πρέπει να υπάρχουν και στους δύο πίνακες). Το αποτέλεσμα περιλαμβάνει πλειάδες των A, B που έχουν ίδια τιμή πάνω σε αυτά τα χαρακτηριστικά.
- Οι παρακάτω εκφράσεις είναι συντακτικά ισοδύναμες

  a Inner Join b USING (c1,c2,c3)

  a Inner Join b ON a.c1=b.c1 AND a.c2=b.c2 AND a.c3=b.c3

Σύνδεση σχέσεων – παραδείγματα

- Σχέση *loan*

loan-number	*branch-name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Σχέση *borrower*

customer-name	*loan-number*
Jones	L-170
Smith	L-230
Hayes	L-155

- Σημείωση: (για τους σκοπούς του παραδείγματος) στη σχέση borrower λείπει πληροφορία για το L-260 και στη σχέση loan λείπει πληροφορία για το L-155

Σύνδεση σχέσεων – Παραδείγματα (1)

- *loan* **inner join** *borrower* **on**
  loan.loan-number = borrower.loan-number

loan-number	*branch-name*	*amount*	*customer-name*	*loan-number*
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230

- *loan* **left outer join** *borrower* **on**
  loan.loan-number = borrower.loan-number

loan-number	*branch-name*	*amount*	*customer-name*	*loan-number*
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-260	Perryridge	1700	*null*	*null*

Σύνδεση σχέσεων – Παραδείγματα (2)

- *loan* **natural inner join** *borrower*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- *loan* **natural right outer join** *borrower*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	null	null	Hayes

Σύνδεση σχέσεων – Παραδείγματα (3)

- *loan* **full outer join** *borrower* **using** *(loan-number)*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	null	null	Hayes

- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Θεωρήσεις (views)

- Παρέχουν ένα μηχανισμό απόκρυψης προκαθορισμένων δεδομένων από προκαθορισμένους χρήστες. Για τη δημιουργία μιας θεώρησης χρησιμοποιούμε την εντολή:

```
CREATE VIEW <view_name>
(<column_name1>, <column_name2>, …, <column_nameN>)
AS
<SQL_select_statement>
```

όπου:

- < SQL_select_statement> είναι οποιαδήποτε έγκυρη έκφραση SQL
- view_name το όνομα της θεώρησης

Παραδείγματα

- Μια θεώρηση που αποτελείται από τα υποκαταστήματα και τους πελάτες τους

```
create view all-customer as
  (select branch-name, customer-name from depositor, account
   where depositor.account-number = account.account-number)
union
  (select branch-name, customer-name from borrower, loan
  where borrower.loan-number = loan.loan-number)
```

- Παράδειγμα: «Βρες τους πελάτες του υποκαταστήματος Perryridge»

```
select customer-name
from all-customer
where branch-name = 'Perryridge'
```

Παραγόμενες σχέσεις

- Παράδειγμα: «Βρες το μέσο υπόλοιπο των λογαριασμών των υποκαταστημάτων τα οποία έχουν λογαριασμούς με μέσο υπόλοιπο άνω των 1200»

```
select branch-name, avg-balance
from (select branch-name, avg (balance)
      from account
      group by branch-name)
     as result (branch-name, avg-balance)
where avg-balance > 1200
```

- **Σημείωση**: δεν χρειαζόμαστε την πρόταση **having**, αφού υπολογίζουμε την προσωρινή σχέση (θεώρηση) *result* στην πρόταση **from** και τα χαρακτηριστικά της *result* μπορούν να χρησιμοποιηθούν απευθείας στην πρόταση **where**.

Το τμήμα With

- Το τμήμα **With** επιτρέπει των ορισμό των θεωρήσεων τοπικά στο ερώτημα, παρά καθολικά. Είναι ουσιαστικά ανάλογο με τη χρήση των διαδικασιών σε μια γλώσσα προγραμματισμού.
- Παράδειγμα: «Βρες τους λογαριασμούς που έχουν το μέγιστο υπόλοιπο»

```
with max-balance(value) as
   select max (balance) from account
select account-number
from account, max-balance
where account.balance = max-balance.value
```

Προσοχή: κάποια εμπορικά ΣΔΒΔ (π.χ. MySQL) δεν υποστηρίζουν WITH

Σύνθετα ερωτήματα με χρήση του With

- Παράδειγμα: «Βρες τα υποκαταστήματα όπου το συνολικό υπόλοιπο των λογαριασμών είναι μεγαλύτερο από το μέσο όρο των συνολικών καταθέσεων σε όλα τα υποκαταστήματα»

```
with branch-total (branch-name, value) as
  select branch-name, sum (balance) from account
  group by branch-name
with branch-total-avg(value) as
  select avg (value) from branch-total
select branch-name
from branch-total, branch-total-avg
where branch-total.value >= branch-total-avg.value
```


- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Τροποποίηση της ΒΔ – Διαγραφή (1)

- Παράδειγμα: «Να διαγραφούν όλοι οι λογαριασμοί από όλα τα υποκαταστήματα που βρίσκονται στην πόλη Needham»

```
delete from account
where branch-name in (select branch-name
                      from branch
                      where branch-city = 'Needham')
delete from depositor
where account-number in
        (select account-number
           from branch, account
           where branch-city = 'Needham'
            and branch.branch-name = account.branch-name)
```


Τροποποίηση της ΒΔ – Διαγραφή (2)

- Παράδειγμα: «Να διαγραφούν οι εγγραφές των λογαριασμών με υπόλοιπο κάτω του μέσου υπολοίπου της τράπεζας»

```
delete from account
where balance < (select avg (balance)
                 from account)
```

  - Πρόβλημα: καθώς διαγράφουμε πλειάδες από τη σχέση *deposit,* το μέσο υπόλοιπο αλλάζει !!
  - Λύση που χρησιμοποιείται στην SQL:

1. Πρώτα υπολογίζουμε το **avg** balance και βρίσκουμε τις πλειάδες που πρέπει να διαγραφούν
2. Έπειτα διαγράφουμε όλες τις πλειάδες που βρήκαμε στο βήμα 1 (χωρίς να υπολογίζουμε εκ νέου το **avg** ή να επανελέγχουμε τις πλειάδες)

Τροποποίηση της ΒΔ – Εισαγωγή (1)

- Παράδειγμα: «Να εισαχθεί μια νέα πλειάδα στη σχέση *account* με κωδικό λογαριασμού 'A-9732', όνομα υποκαταστήματος 'Perryridge' και υπόλοιπο 1200»

  insert into *account*
  values ('A-9732', 'Perryridge',1200)

  ή ισοδύναμα

  insert into *account (branch-name, balance, account-number)*
  values ('Perryridge', 1200, 'A-9732')

- Αν δεν γνωρίζουμε την τιμή ενός χαρακτηριστικού ...

  insert into *account*
  values ('A-777','Perryridge', *null*)

Τροποποίηση της ΒΔ – Εισαγωγή (2)

- Παράδειγμα: «Να δοθεί ως δώρο σε όλους τους δανειολήπτες του υποκαταστήματος Perryridge ένας καταθετικός λογαριασμός με ποσό 200. Ο λογαριασμός αυτός ως κωδικό τον κωδικό του αντίστοιχου δανείου»

  insert into *account*
  select *loan-number, branch-name,* 200 **from** *loan*
  where *branch-name* = 'Perryridge'
  insert into *depositor*
  select *customer-name, loan-number* **from** *loan, borrower*
  where branch-name = 'Perryridge'
  and *loan.account-number = borrower.account-number*

- Η έκφραση select-from-where αποτιμάται πλήρως πριν από την εισαγωγή στη σχέση οποιουδήποτε εκ των αποτελεσμάτων της

Τροποποίηση της ΒΔ – Ενημέρωση

- Παράδειγμα: «Να δοθεί αύξηση 6% (5%) σε όλους τους λογαριασμούς με υπόλοιπο μεγαλύτερο των (μέχρι) 10.000»
  - Γράφουμε δυο εντολές **update** (η σειρά είναι σημαντική):

```
update account
set balance = balance * 1.06 where balance > 10000

update account
set balance = balance * 1.05 where balance ≤ 10000
```

  - Μπορεί να υλοποιηθεί με μία εντολή, με χρήση της εντολής **case**:

```
update account
set balance =  case
    when balance <= 10000 then balance *1.05
   else   balance * 1.06
end
```


Παράδειγμα ενημέρωσης μιας θεώρησης

- Δημιουργήστε μια θεώρηση για τα δεδομένα της σχέσης *loan*, κρύβοντας το χαρακτηριστικό *amount*

```
create view branch-loan as
        select branch-name, loan-number
        from loan
```

- Προσθέστε μια νέα πλειάδα στη θεώρηση *branch-loan*

```
insert into branch-loan
        values ('Perryridge', 'L-307')
```

  Αυτή η εισαγωγή πρέπει να αναπαρασταθεί με την εισαγωγή της πλειάδας

  ('L-307', 'Perryridge', *null*)

  στη σχέση *loan*
- Οι ενημερώσεις σε πιο πολύπλοκες θεωρήσεις είναι δύσκολο ή αδύνατο να μεταφραστούν και επομένως δεν επιτρέπονται.
  - Οι πιο πολλές υλοποιήσεις της SQL επιτρέπουν ενημερώσεις μόνο σε απλές θεωρήσεις (χωρίς συναθροίσεις) που ορίζονται σε μια μόνο σχέση

- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Γλώσσα Ορισμού Δεδομένων

- Η γλώσσα SQL χρησιμοποιείται και ως **Γλώσσα Ορισμού Δεδομένων** (Data Definition Language - DDL), με χρήση της οποίας καθορίζονται οι σχέσεις της ΒΔ, αλλά και επιπλέον πληροφορία για κάθε σχέση, όπως:
  - Το σχήμα για κάθε σχέση
  - Το πεδίο τιμών κάθε χαρακτηριστικού
  - Περιορισμούς ακεραιότητας
  - Το σύνολο των ευρετηρίων που συντηρούνται για κάθε σχέση (*)
  - Τη δομή φυσικής αποθήκευσης κάθε σχέσης στο δίσκο (*)

(*) θέματα που θα μελετηθούν σε επόμενη ενότητα

Τύποι πεδίου ορισμού στην SQL

- **char(n).** Σταθερού μήκους συμβολοσειρά, με μήκος *n* (το *n* καθορίζεται από το χρήστη).
- **varchar(n).** Μεταβλητού μήκους συμβολοσειρά, με μέγιστο μήκος *n* (το *n* καθορίζεται από το χρήστη).
- **int.** Ακέραιος (το εύρος του εξαρτάται από τη μηχανή).
- **smallint.** Μικρός ακέραιος (υποσύνολο των ακεραίων, το εύρος του εξαρτάται από τη μηχανή).
- **numeric(p,d).** Αριθμός σταθερής υποδιαστολής, με ακρίβεια *p* ψηφίων (το *p* καθορίζεται από το χρήστη), με *n* ψηφία στα δεξιά της υποδιαστολής.
- **real, double precision.** Αριθμός κινητής υποδιαστολής απλής και διπλής ακρίβειας, αντίστοιχα (η ακρίβειά του εξαρτάται από τη μηχανή).
- **float(n).** Αριθμός κινητής υποδιαστολής. Ο χρήστης καθορίζει την ακρίβεια τουλάχιστον *n* ψηφίων.

Τύποι Date/Time στην SQL

- **date.** Ημερομηνίες, που περιέχουν τετραψήφιο έτος, μήνα και ημέρα
  - Π.χ. **date** '2001-7-27'
- **time.** Ώρα της ημέρας (ώρες, λεπτά, δευτερόλεπτα).
  - Π.χ. **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: ημερομηνία + ώρα
  - Π.χ. **timestamp** '2001-7-27 09:00:30.75'
- **Interval**: χρονική περίοδος
  - Π.χ. Interval '1' day
  - Η αφαίρεση μιας τιμής date / time / timestamp από μια άλλη δίνει μια interval τιμή
  - Οι interval τιμές μπορούν να προστεθούν σε τιμές date / time / timestamp

Σύνταξη Create Table

- Μια σχέση ορίζεται στην SQL με την εντολή **create table**:

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (περιορισμός ακεραιότητας$_1$),
  ...,
  (περιορισμός ακεραιότητας$_k$))

  - *r* είναι το όνομα της σχέσης
  - κάθε A_i είναι ένα όνομα χαρακτηριστικού στο σχήμα της σχέσης *r*
  - D_i είναι ο τύπος δεδομένων των τιμών στον τύπο πεδίου ορισμού του χαρακτηριστικού A_i
- Παράδειγμα:

```
create table branch
    (branch-name   char(15) not null,
     branch-city   char(30),
     assets        integer)
```


Οι εντολές Drop και Alter Table

- Η εντολή **drop table** διαγράφει όλες τις πληροφορίες για τη σχέση που διαγράφεται από τη βάση δεδομένων.
- Η εντολή **alter table** χρησιμοποιείται για την προσθήκη χαρακτηριστικών σε μια σχέση που έχει ήδη δημιουργηθεί. Σε όλες τις πλειάδες στη σχέση καταχωρείται η τιμή *null* για το νέο χαρακτηριστικό. Η μορφή της εντολής **alter table** είναι:

  alter table *r* **add** *A D*

  όπου *A* είναι το όνομα του χαρακτηριστικού που προστίθεται στη σχέση *r* και *D* είναι ο τύπος πεδίου ορισμού του *A*.
- Η εντολή **alter table** μπορεί επίσης να χρησιμοποιηθεί για τη διαγραφή χαρακτηριστικών από μια σχέση

  alter table *r* **drop** *A*

  όπου *A* είναι το όνομα ενός χαρακτηριστικού της σχέσης *r*
  - **Σημείωση:** πολλά ΣΔΒΔ δεν υποστηρίζουν διαγραφή χαρακτηριστικών

- Βασική δομή
- Συνήθεις τύποι ερωτημάτων
- Συνδέσεις (joins)
- Θεωρήσεις (views) και παραγόμενες σχέσεις
- Τροποποίηση της βάσης δεδομένων
- Η SQL ως γλώσσα ορισμού δεδομένων
- Ακεραιότητα δεδομένων

Απλοί περιορισμοί ακεραιότητας

- **not null**
- **primary key** (A_1, ..., A_n)
- **check** *(P)*, όπου *P* είναι ένα κατηγόρημα

  Παράδειγμα: Ορίστε το *branch-name* ως το πρωτεύον κλειδί για τη σχέση *branch* και εξασφαλίστε ότι οι τιμές του χαρακτηριστικού *assets* είναι μη αρνητικές.

```
create table branch
(branch-name    char(15),
branch-city     char(30)
assets  integer,
primary key (branch-name),
check (assets >= 0))
```

- **Σημείωση:** Στην SQL-92 η δήλωση ενός χαρακτηριστικού ως **primary key** αυτόματα εγγυάται ότι δεν επιτρέπεται να πάρει **null τιμές**

Περιορισμοί πεδίου

- Η πρόταση **check** στην SQL-92 επιτρέπει να διατυπώνουμε περιορισμούς για τα πεδία:
  - Π.χ. η πρόταση check εξασφαλίζει ότι το πεδίο ωρομίσθιο *(hourly-wage)* επιτρέπει μόνο τιμές μεγαλύτερες από μία συγκεκριμένη τιμή.

    create domain *hourly-wage* numeric(5,2)
    constraint *value-test* check*(value* > = 4.00)
  - Το πεδίο έχει έναν περιορισμό που εξασφαλίζει ότι το ωρομίσθιο είναι μεγαλύτερο ή ίσο του 4.00
  - Η πρόταση constraint *value-test* είναι προαιρετική.
- Στον έλεγχο του πεδίου μπορεί να υπάρχουν πολύπλοκες συνθήκες:
  - create domain *AccountType* char(10)
    constraint *account-type* check (value in ('checking', 'saving'))
  - check (*branch-name* in (select *branch-name* from *branch*))

Αναφορική ακεραιότητα στην SQL

- Το πρωτεύον κλειδί και τα ξένα κλειδιά μπορούν να οριστούν ως μέρος της SQL δήλωσης create table:
  - Ο όρος **primary key** της δήλωσης **create table** περιλαμβάνει τη λίστα των γνωρισμάτων που συνθέτουν το πρωτεύον κλειδί.
  - Ο όρος **foreign key** της δήλωσης **create table** περιλαμβάνει και μια λίστα με τα γνωρίσματα που συνθέτουν το ξένο κλειδί και το όνομα της σχέσης στην οποία αναφέρεται το ξένο κλειδί.

Αναφορική ακεραιότητα στην SQL (συν.)

```
create table customer
    (customer-name      char(20),
     customer-street    char(30),
     customer-city      char(30),
     primary key (customer-name))

create table branch
    (branch-name    char(15),
     branch-city    char(30),
     assetsinteger,
     primary key (branch-name))

create table account
    (account-number     char(10),
     branch-name        char(15),
     balance            integer,
     primary key (account-number),
     foreign key (branch-name) references branch)

create table depositor
    (customer-name      char(20),
     account-number     char(10),
     primary key (customer-name, account-number),
     foreign key (account-number) references account,
     foreign key (customer-name) references customer)
```


Συμπαρασυρόμενες *(cascading)* ενέργειες

```
create table account
    . . .
    foreign key(branch-name)  references branch
                              on delete cascade
                              on update cascade
    . . . )
```

- Λόγω του όρου **on delete cascade**, αν η διαγραφή μιας πλειάδας της σχέσης *branch* οδηγήσει σε παραβίαση του περιορισμού αναφορικής ακεραιότητας, τότε η διαγραφή αυτή θα "μεταφερθεί" διαδοχικά και στη σχέση *account*, διαγράφοντας τις πλειάδες που αναφέρονται στο υποκατάστημα που διαγράφηκε από τη σχέση account.
- Κατά παρόμοιο τρόπο εκτελούνται οι συμπαρασυρόμενες ενημερώσεις (**on update cascade**).

Συμπαρασυρόμενες ενέργειες (συν.)

- Αν υπάρχει μία αλυσίδα από εξαρτήσεις ξένων κλειδιών μεταξύ πολλαπλών σχέσεων, με τη δήλωση **on delete cascade** που καθορίζεται για κάθε εξάρτηση, μια διαγραφή ή τροποποίηση στο ένα άκρο της αλυσίδας μπορεί να διαδοθεί κατά μήκος ολόκληρης της αλυσίδας.
- Αν οι συμπαρασυρόμενες διαγραφές οδηγήσουν σε παραβίαση του περιορισμού που δεν μπορεί να αντιμετωπιστεί από κάποια περαιτέρω συμπαρασυρόμενη λειτουργία, το σύστημα εμποδίζει την δοσοληψία *(transaction).* Ως αποτέλεσμα, όλες οι αλλαγές που προκλήθηκαν από τη δοσοληψία και τις διαδοχικές της ενέργειες αναιρούνται.

Διαβεβαιώσεις (Assertions)

- Η **διαβεβαίωση** (**assertion**) είναι ένα κατηγόρημα που εκφράζει μια συνθήκη που θέλουμε να ικανοποιεί πάντα η βάση δεδομένων.
- Ένα assertion στην SQL έχει τη μορφή:

  create assertion <assertion-name> **check** <predicate>
- Όταν δημιουργείται ένα assertion, το σύστημα ελέγχει την ορθότητά του και επαναλαμβάνεται ο έλεγχος σε κάθε ενημέρωση *(update)* που μπορεί να οδηγήσει σε παραβίαση του assertion.
  - Ο έλεγχος αυτός μπορεί να εισάγει μια σημαντική επιβάρυνση στο χρόνο απόκρισης. Συνεπώς τα assertions πρέπει να χρησιμοποιούνται με μεγάλη προσοχή.
- Η ικανοποίηση του assertion : $\forall$ X: P(X)
  επιτυγχάνεται μέσω της συνθήκης: $\nexists$ X: $\neg$ P(X)
  - Σε αντίθεση με την πρώτη, η δεύτερη συνθήκη μπορεί να υλοποιηθεί στην SQL.

Παράδειγμα assertion -1

- Το άθροισμα των δανείων κάθε υποκαταστήματος θα πρέπει να είναι μικρότερο από το άθροισμα των υπολοίπων των λογαριασμών του εν λόγω υποκαταστήματος.

```
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)))
```

$\forall$ branch: sum(loan.amount) < sum(account.balance)

$\Rightarrow \nexists$ branch: sum(loan.amount) ≥ sum(account.balance)

Παράδειγμα assertion -2

- Κάθε δάνειο θα πρέπει να έχει έναν τουλάχιστον δανειολήπτη που να διατηρεί λογαριασμό με ελάχιστο υπόλοιπο 1000.

```
create assertion balance-constraint check
    (not exists (
        select * from loan l
        where not exists (
            select *
            from borrower b, depositor d, account a
            where l.loan-number = b.loan-number
                  and b.customer-name = d.customer-name
                  and d.account-number = a.account-number
                  and a.balance >= 1000)))
```

$\forall$ loan: $\exists$ borrower with balance ≥ 1000

$\Rightarrow \nexists$ loan: $\exists$ borrower with balance ≥ 1000

Ενεργοποιητές (Triggers)

- Ένας **ενεργοποιητής** (**trigger**) είναι μια εντολή που εκτελείται αυτόματα από το σύστημα ως άμεσο αποτέλεσμα της τροποποίησης της βάσης.
- Ο σχεδιασμός ενός μηχανισμού trigger απαιτεί:
  - Τον καθορισμό των συνθηκών κάτω από τις οποίες θα «πυροδοτείται» ο trigger.
  - Τον καθορισμό των ενεργειών που θα πραγματοποιούνται κατά την εκτέλεση του trigger.
- Ο μηχανισμός των triggers εισήχθη ως SQL standard στην SQL:1999, αλλά από νωρίτερα υπήρχε σχετική υποστήριξη από τις περισσότερες βάσεις δεδομένων.

Παράδειγμα trigger

- Ας υποθέσουμε πως αντί να επιτρέπει αρνητικά υπόλοιπα στους λογαριασμούς, η τράπεζα αντιμετωπίζει τις υπεραναλήψεις *(overdrafts)* ως εξής:
  - Θέτει το υπόλοιπο του λογαριασμού στο μηδέν
  - Δημιουργεί ένα δάνειο με το ακάλυπτο ποσό
  - Δίνει στο δάνειο έναν αριθμό δανείου παρόμοιο με τον αριθμό λογαριασμού από τον οποίο προέρχεται το ακάλυπτο ποσό.
- Η συνθήκη "πυροδότησης" του trigger είναι κάποια ενημέρωση στη σχέση *account* που οδηγεί σε αρνητικό υπόλοιπο.

Παράδειγμα trigger (SQL:1999)

```
create trigger overdraft-trigger after update on account
referencing new row as nrow
for each row
when nrow.balance < 0
begin atomic
      insert into borrower
         (select customer-name, account-number
          from depositor
          where nrow.account-number =
                depositor.account-number);
      insert into loan values
         (nrow.account-number, nrow.branch-name,
                                   – nrow.balance);
      update account set balance = 0
         where account.account-number = nrow.account-number
end
```


Πότε πυροδοτούνται οι triggers

- Υπάρχουν τρία συμβάντα που μπορούν να πυροδοτήσουν έναν trigger: **insert**, **delete** και **update.**
- Στην περίπτωση των ενημερώσεων οι triggers μπορούν να περιοριστούν σε συγκεκριμένα γνωρίσματα
  - **Π.χ. create trigger** *overdraft-trigger* **after update of** *balance on account*
- Οι τιμές των γνωρισμάτων είναι προσπελάσιμες πριν και μετά από κάθε ενημέρωση.
  - **referencing old row as ...** (για τις διαγραφές και τις ενημερώσεις)
  - **referencing new row as ...** (για τις εισαγωγές και τις ενημερώσεις)

Πότε πυροδοτούνται οι triggers (συν.)

- Οι triggers μπορούν να πυροδοτηθούν πριν από κάποιο συμβάν, γεγονός που μπορεί να χρησιμοποιηθεί ως επιπλέον περιορισμός.
- π.χ. μετατροπή των κενών ‘ ‘ σε null.

```
create trigger setnull-trigger before update on r
referencing new row as nrow
for each row
   when nrow.phone-number = ‘ ‘
   set nrow.phone-number = null
```


Triggers σε επίπεδο εντολής SQL

- Αντί να εκτελείται μια ξεχωριστή ενέργεια για κάθε γραμμή (row) που επηρεάζεται, μπορεί να εκτελείται μία ξεχωριστή ενέργεια σε επίπεδο εντολής (statement), για όλες τις γραμμές που επηρεάζονται από μια δοσοληψία. Στην περίπτωση αυτή:
  - χρησιμοποιείται η εντολή **for each statement** αντί της εντολής **for each row.**
  - Χρησιμοποιείται η δήλωση **referencing old table** ή **referencing new table** προκειμένου να αναφερθούμε σε προσωρινούς πίνακες που περιέχουν τις γραμμές που επηρεάζονται.
  - Ο τρόπος αυτός είναι πιο αποδοτικός σε περιπτώσεις εντολών SQL που ενημερώνουν ένα μεγάλο πλήθος γραμμών.

Εξωτερικές ενέργειες

- Μερικές φορές κατά την ενημέρωση της βάσης δεδομένων απαιτείται η ενεργοποίηση κάποιων εξωτερικών ενεργειών, όπως η εκ νέου παραγγελία ενός προϊόντος όταν η ποσότητά του πέσει κάτω από μία συγκεκριμένη τιμή ή η ενεργοποίηση ενός διακόπτη.
- Οι triggers προφανώς δεν μπορούν να χρησιμοποιηθούν για την κατευθείαν υλοποίηση εξωτερικών ενεργειών, ΑΛΛΑ
  - μπορούν να χρησιμοποιηθούν για την καταγραφή σε έναν ξεχωριστό πίνακα των ενεργειών που πρέπει να ληφθούν, και
  - μπορούμε να έχουμε μια εξωτερική διεργασία που θα σαρώνει συνεχώς τον πίνακα και θα διεκπεραιώνει τις εξωτερικές ενέργειες.

Εξωτερικές ενέργειες (συν.)

- Π.χ. μια ΒΔ για μια αποθήκη προϊόντων, με τους ακόλουθους πίνακες
  - *inventory(item, level):* η ποσότητα κάθε αντικειμένου στην αποθήκη
  - *minlevel(item, level):* το ελάχιστο επιθυμητό επίπεδο ανά αντικείμενο
  - *reorder(item, amount):* η ποσότητα που πρέπει να παραγγελθεί
  - *orders(item, amount):* οι παραγγελίες που πρέπει να τοποθετηθούν στον πίνακα των ενεργειών (θα διαβαστούν από την εξωτερική διεργασία)

```
create trigger reorder-trigger after update of level on inventory
referencing old row as orow, new row as nrow
for each row
    when nrow.level < =
            (select level  from minlevel m   where m.item = orow.item)
      and orow.level >
            (select level from minlevel m   where m.item = orow.item)
    begin
            insert into orders
              (select item, amount  from reorder  where reorder.item = orow.item)
    end
```