Η Γλώσσα SQL

Η "standard" γλώσσα για σχεσιακές βάσεις δεδομένων.

Αρχικά Sequel στην IBM ως μέρος του System R, τώρα SQL (Stuctured Query Language)

SQL–89, SQL–92

Η SQL έχει διάφορα τμήματα:

- Γλώσσα Ορισμού Δεδομένων (ΓΟΔ)
- Γλώσσα Χειρισμού Δεδομένων (ΓΧΔ)
- Ενσωματωμένη Γλώσσα Χειρισμού Δεδομένων
- Ορισμό ΄Οψεων
- Εξουσιοδότηση (authentication)
- Ακεραιότητα
- ΄Ελεγχο Συναλλαγών

Η Γλώσσα SQL

1. Βασική Δομή
2. Πράξεις σε Σύνολα
3. Συναθροιστικές Συναρτήσεις
4. Η Τιμή null
5. Φωλιασμένες Υποερωτήσεις
6. Παραγόμενες Σχέσεις
7. Σχέσεις Συνένωσης
8. Γλώσσα Ορισμού Δεδομένων
9. Φωλιασμένη SQL
10. Άλλα Χαρακτηριστικά

Βασική Δομή

Μια χαρακτηριστική ερώτηση σε SQL έχει την εξής μορφή:

select A_1, A_2, $\ldots$, A_n
from R_1, R_2, $\ldots$, R_m
where P

όπου A_i είναι γνωρίσματα, τα R_i σχέσεις και το P κατηγόρημα (συνθήκη).

Η βασική δομή μιας SQL έκφρασης αποτελείται από 3 προτάσεις:

- **select:** αντιστοιχεί στην πράξη της προβολής της σχεσιακής άλγεβρας. Χρησιμοποιείται για να απαριθμήσει τα γνωρίσματα που θέλουμε να υπάρχουν στο αποτέλεσμα της ερώτησης.
- **from:** αντιστοιχεί στην πράξη του καρτεσιανού γινομένου της σχεσιακής άλγεβρας. Απαριθμεί ποιες σχέσεις θα χρησιμοποιηθούν για τον υπολογισμό του αποτελέσματος.
- **where:** αντιστοιχεί στη συνθήκη της πράξης της επιλογής στη σχεσιακή άλγεβρα. Το κατηγόρημα έχει γνωρίσματα των σχέσεων που εμφανίζονται στο **from**

Βασική Δομή (συνέχεια)

Η SQL δημιουργεί το καρτεσιανό γινόμενο των σχέσεων στο **from**, εκτελεί μια επιλογή με βάση το κατηγόρημα του **where** και προβάλει το αποτέλεσμα με βάση τα γνωρίσματα του **select**.

Συχνά μετατρέπει την έκφραση σε μια ισοδύναμη της βέλτιστη.

Ισοδυναμεί με την έκφραση:

$$\Pi_{A_1,A_2,\ldots,A_n}(\sigma_P(R_1 \times R_2 \times \ldots \times R_m))$$

Όταν δεν υπάρχει το **where**, το P θεωρείται ότι ισχύει.

Δε γίνεται απαλοιφή των διπλών εμφανίσεων.

Παράδειγμα: *Ονόματα των πελατών που μένουν στην πόλη Chicago*

> **select** *customer–name*
> **from** *customer*
> **where** *customer–city* = "Chicago"

Select

Παράδειγμα: *Ονόματα όλων των υποκαταστημάτων που έχουν δώσει δάνεια*

> **select** *branch–name*
> **from** *loan*

Η SQL επιτρέπει πολλαπλές εμφανίσεις της ίδιας πλειάδας σε μια σχέση. Μια σχέση στην SQL είναι ένα πολυσύνολο (multiset) ή θύλακας (bag).

Απαλοιφή διπλών εμφανίσεων:

> **select distinct** *branch–name*
> **from** *loan*

Επίσης, **select all**

Όλα τα γνωρίσματα:

> **select ***
> **from** *loan*

Αριθμητικές πράξεις $(+, -, *, /)$ ανάμεσα σε σταθερές ή γνωρίσματα πλειάδων.

> **select** *branch–name*, *loan–number*, *amount* * 100
> **from** *loan*

Επιστρέφει μια σχέση ίδια με τη σχέση loan μόνο που το γνώρισμα amount έχει πολλαπλασιαστεί με το 100.

Where

Παράδειγμα: *Τους αριθμούς όλων των δανείων που έγιναν στο υποκατάστημα Perryridge με ποσό μεγαλύτερο των $1200*

> **select** *loan–number*
> **from** *loan*
> **where** *branch-name* = "Perryridge" **and** *amount* > 1200

Λογικοί τελεστές: **and**, **or**, **not**. Τελεστές σύγκρισης: <, <=, >, >=, =, <>, **between**, **not between** ανάμεσα σε αριθμητικές εκφράσεις, συμβολοσειρές (strings), και ειδικούς τύπους.

Παράδειγμα με τον τελεστή **between**

Παράδειγμα:

> **select** *loan–number*
> **from** *loan*
> **where** *amount* **between** 80000 **and** 100000

αντί του:

> **select** *loan–number*
> **from** *loan*
> **where** *amount* >= 80000 **and** *amount* <= 100000

From

Όταν το ίδιο γνώρισμα εμφανίζεται στο σχήμα περισσότερων από μια σχέσεων, τότε διάκριση βάση του συμβολισμού:

όνομα-σχέσης.όνομα-γνωρίσματος

Παράδειγμα φυσικής συνένωσης: *Τα ονόματα και τους αριθμούς δανείων όλων των πελατών που έχουν πάρει δάνειο από την τράπεζα*

> **select distinct** *customer.name*, *borrower.loan–number*
> **from** *borrower*, *loan*
> **where** *borrower.loan–number* = *loan.loan–number*

Παράδειγμα: (*επιπρόσθετος περιορισμός το δάνειο να είναι από το υποκατάστημα στο Perryridge*):

> **select distinct** *customer.name*, *borrower.loan–number*
> **from** *borrower*, *loan*
> **where** *borrower.loan–number* = *loan.loan–number*
>   **and** *branch–name* = "Perryridge"

Αλλαγή Ονόματος

Τα ονόματα των γνωρισμάτων στο αποτέλεσμα είναι αυτά των σχέσεων στην ερώτηση. Δυνατότητα αλλαγής του ονόματος τόσο μιας σχέσης όσο και ενός γνωρίσματος:

παλιό-όνομα **as** *νέο-όνομα*

Το **as** μπορεί να εμφανίζεται στο **select** ή στο **from**:

Για παράδειγμα:

select distinct *customer.name*,
borrower.loan–number **as** *loan–id*
from *borrower*, *loan*
where *borrower.loan–number* = *loan.loan–number*

Χρήσιμο όταν (α) δυο σχέσεις του **from** έχουν γνωρίσματα με το ίδιο όνομα, (β) όταν έχουμε αριθμητικές εκφράσεις στο **select** και δεν έχουν όνομα, (γ) όταν θέλουμε να αλλάξουμε το όνομα του γνωρίσματος στο αποτέλεσμα.

Μεταβλητές Πλειάδων

Μια μεταβλητή πλειάδας μπορεί να οριστεί στο **from** χρησιμοποιώντας το **as:**

> **select distinct** *T.name*, *S.loan–number*
> **from** *borrower* **as** *T*, *loan* **as** *S*
> **where** *T.loan–number* = *S.loan–number*

Οι μεταβλητές είναι ιδιαίτερα χρήσιμες όταν θέλουμε να συγκρίνουμε δυο πλειάδες τις ίδιας σχέσης.

Παράδειγμα: *Τα ονόματα όλων των υποκαταστημάτων που έχουν καταθέσεις μεγαλύτερες από τις καταθέσεις ενός τουλάχιστον υποκαταστήματος του Brooklyn.*

> **select** branch–name
> **from** *branch* **as** *S*, *branch* **as** *T*
> **where** *T.assets* > *S.assets* **and**
> S.branch–name = *“Brooklyn”*

Πράξεις με Συμβολοσειρές

Η πιο συνηθισμένη πράξη είναι ταίριασμα προτύπων:

- % ταιριάζει οποιαδήποτε συμβολοσειρά
- _ ταιριάζει οποιοδήποτε χαρακτήρα

Γίνεται διάκριση ανάμεσα σε κεφαλαία και μικρά

Σύγκριση χρησιμοποιώντας το **like**, **not like**

Παράδειγμα: *Τα ονόματα όλων των πελατών που μένουν σε οδό της οποίας το όνομα περιέχει την υποσυμβολοσειρά "Main"*

> **select** *customer–name*
> **from** *customer*
> **where** *customer–street* **like**

Πολλές ακόμα πράξεις είναι διαθέσιμες.

Διάταξη των Πλειάδων

Χρήση του **order by** ώστε οι πλειάδες στο αποτέλεσμα να είναι ταξινομημένες με βάση το αντίστοιχο γνώρισμα

> **select distinct** *customer–name*, *borrower.loan–number*
> **from** *borrower*, *loan*
> **where** *borrower.loan–number* = *loan.loan–number*
>   **and** *branch–name* = “Perryridge”
> **order by** *customer–name*

Default: αύξουσα διάταξη, αλλά και άμεσα χρησιμοποιώντας το **asc** (αύξουσα) ή το **desc** (φθήνουσα). Επίσης, ταξινόμηση με βάση πολλά γνωρίσματα,

Παράδειγμα:

> **select** *
> **from** *loan*
> **order by** *amount* **desc**, *loan–number* **asc**

Η ταξινόμηση είναι δαπανηρή λειτουργεία.

Πράξεις σε Σύνολα

Πράξεις: **union** ($\cup$), **intersection** ($\cap$), **except** ($-$), εφαρμόζονται σε συμβατές σχέσεις.

Παράδειγμα **union**: *Τα ονόματα όλων των πελατών που έχουν καταθέσεις ή/και έχουν πάρει δάνειο*

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

Απαλοιφή διπλών εμφανίσεων, εκτός αν χρησιμοποιηθεί το **union all** στη θέση του **union**.

Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Παράδειγμα **intersect**: *Τα ονόματα όλων των πελατών που έχουν καταθέσεις και έχουν πάρει δάνειο*

(**select distinct** *customer–name*
from *depositor*)
intersect
(**select distinct** *customer–name*
from *borrower*)

Αντίστοιχα υπάρχει **intersect all**.

Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Πράξεις σε Σύνολα (συνέχεια)

Παράδειγμα **except**: *Τα ονόματα όλων των πελατών που έχουν καταθέσεις και δεν έχουν πάρει δάνειο*

> (**select distinct** *customer–name*
> **from** *depositor*)
> **except**
> (**select** *customer–name*
> **from** *borrower*)

Αντίστοιχα υπάρχει **except all**.

Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Συναθροιστικές Συναρτήσεις

Η SQL έχει 5 built-in συναθροιστικές συναρτήσεις:

- Μέσος όρος: **avg** (μόνο σε αριθμούς)
- Ελάχιστο: **min**
- Μέγιστο: **max**
- Άθροισμα: **sum** (μόνο σε αριθμούς)
- Πλήθος: **count**

Παράδειγμα: *Μέσο υπόλοιπο όλων των λογαριασμών στο υποκατάστημα Perryridge*

> **select avg**(*balance*)
> **from** *acount*
> **where** *branch–name* = "Perryridge"

Το αποτέλεσμα είναι μια σχέση με ένα γνώρισμα και μια γραμμή, μπορούμε να δώσουμε όνομα στο γνώρισμα χρησιμοποιώντας το **as**

Μπορούμε να εφαρμόσουμε τις συναρτήσεις όχι μόνο σε ένα σύνολο από πλειάδες, αλλά σε ομάδες από σύνολα πλειάδων. Οι ομάδες προσδιορίζονται χρησιμοποιώντας το **group by**

Συναθροιστικές Συναρτήσεις (συνέχεια)

Παράδειγμα: *Μέσο υπόλοιπο των λογαριασμών σε κάθε υποκατάστημα*

> **select** *branch-name*, **avg**(*balance*)
> **from** *acount*
> **group by** *branch–name*

Αν θέλουμε να απαλείψουμε διπλές εμφανίσεις χρησιμοποιούμε τη λέξη-κλειδί **distinct** στην αντίστοιχη έκφραση.

Παράδειγμα: *Αριθμός καταθετών σε κάθε υποκατάστημα*

> **select** *branch-name*, **count**(**distinct** *customer–name*)
> **from** *depositor*, *account*
> **group by** *branch–name*

Για να μετρήσουμε πόσες πλειάδες έχει μια σχέση:

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

Δε μπορούμε να χρησιμοποιήσουμε το **distinct** με το **count**(*).

Μπορούμε να εφαρμόσουμε μια συνθήκη σε μια συγκεκριμένη ομάδα από πλειάδες χρησιμοποιώντας το **having**

Συναθροιστικές Συναρτήσεις (συνέχεια)

Παράδειγμα: *Ονόματα υποκαταστημάτων με μέσο υπόλοιπο μεγαλύτερο των $1200*

> **select** *branch-name*, **avg**(*balance*)
> **from** *account*
> **group by** *branch–name*
> **having avg**(*balance*) > 1200

Η συνθήκη του **having** εφαρμόζεται αφού σχηματιστούν οι ομάδες και υπολογιστούν οι συναθροιστικές συναρτήσεις.

΄Οταν εμφανίζονται και το **where** και το **having**: Η συνθήκη του **where** εφαρμόζεται πρώτα, οι πλειάδες που ικανοποιούν αυτή τη συνθήκη τοποθετούνται σε ομάδες με βάση το **group by** και μετά αν υπάρχει συνθήκη στο **having** εφαρμόζεται στις ομάδες.

Παράδειγμα: *Μέσο υπόλοιπο για κάθε πελάτη που ζεί στο Harrison και έχει τουλάχιστον τρεις λογαριασμούς*

> **select** *depositor.customer–name*, **avg**(*balance*)
> **from** *depositor*, *account*, *customer*
> **where** *depositor.acount–number* = acount.acount–number
> **and** *depositor.customer–name* = customer.customer–name
> **and** *customer–city* = "Harrison"
> **group by** *depositor.customer–name*
> **having count** (**distinct** *depositor.acount–number*) >= 3

Η Τιμή null

Χρήση της λέξης κλειδί **null** (**not null**) σε μια συνθήκη για να ελέξουμε αν μια τιμή είναι null.

> **select** *loan–number*
> **from** *loan*
> **where** *amount* **is null**

Εμφάνιση null

- Σε αριμητικές πράξεις: το αποτέλεσμα είναι null όταν οποιαδήποτε τιμή είναι null
- Σε συγκρίσεις: σύγκριση με null συνήθως δίνει αποτέλεσμα **false**
- Σε συναθροιστικές συναρτήσεις: αγνοείται πλην από το **count(*)**
  Παράδειγμα:
  > **select sum**(*amount*)
  > **from** *loan*

Φωλιασμένες Υποερωτήσεις

Η SQL επιτρέπει το φώλιασμα υποερωτήσεων.

Μια υποερώτηση είναι μια έκφραση **select-from-where** που χρησιμοποείται μέσα σε μια άλλη ερώτηση.

1. Ο τελεστής in

Ο τελεστής **in** (**not in**) ελέγχει αν μια πλειάδα ανήκει σε ένα σύνολο από πλειάδες που έχουν προκύψει από μια έκφραση **select-from-where**.

Παραδείγματα:

(1) *Τα ονόματα όλων των πελατών που έχουν πάρει δάνειο και έχουν καταθέσεις*

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

Φωλιασμένες Υποερωτήσεις (συνέχεια)

(2) *Τα ονόματα όλων των πελατών που έχουν πάρει δάνειο και έχουν καταθέσεις στο υποκατάστημα Perryridge*

> **select distinct** *customer–name*
> **from** *borrower*, *loan*
> **where** *borrower.loan–number* = *loan.loan–number*
>   **and** *branch–name* = "Perryridge"
>   **and** (*branch–name*, *customer–name*) **in**
>   (**select** *branch–name*, *customer–name*
>   **from** *depositor*, *account*
>   **where** *depositor.account–number* =
>   *account.account–number*)

Μπορεί να χρησιμοποιηθούν και με enumerated σύνολα

Παράδειγμα: *Τα ονόματα όλων των πελατών που έχουν πάρει δάνειο και δε λέγονται "Smith" ή "Jones".*

> **select distinct** *customer–name*
> **from** *borrower*
> **where** *customer–name* **not in** ("Smith", "Jones")

2. Σύγκριση Συνόλων

(i) Ο τελεστής some (any)

έχει τη σημασία του τουλάχιστον ένα από ένα σύνολο

Φωλιασμένες Υποερωτήσεις (συνέχεια)

Παράδειγμα: *Τα ονόματα όλων των υποκαταστημάτων που έχουν καταθέσεις μεγαλύτερες από τις καταθέσεις ενός τουλάχιστον υποκαταστήματος του Brooklyn.*

select distinct *branch–name*
from *branch*
where *assets* > **some** (**select** *assets*
from *branch*
where *branch–name* = "Brooklyn")

επίσης: < **some**, <= **some**, >= **some**, = **some**, <> **some**

(ii) Ο τελεστής all

έχει τη σημασία από όλα τα στοιχεία ενός συνόλου

Παράδειγμα: *Τα ονόματα όλων των υποκαταστημάτων που έχουν καταθέσεις μεγαλύτερες από τις καταθέσεις κάθε υποκαταστήματος του Brooklyn.*

select distinct *branch–name*
from *branch*
where *assets* > **all** (**select** *assets*
from *branch*
where *branch–name* = "Brooklyn")

Φωλιασμένες Υποερωτήσεις (συνέχεια)

επίσης: < **all**, <= **all**, >= **all**, = **all**, <> **all**

Παράδειγμα: *Το υποκατάστημα με το μεγαλύτερο μέσο υπόλοιπο.*

```
select distinct branch–name
from account
group by branch–name
having avg (balance) >= all (select avg  (balance)
                             from account
                             group by branch–name)
```

3. Έλεγχος για Άδεια Σχέση

Ο τελεστής exists: επιστρέφει **true** ανν η υποερώτηση δεν είναι κενή

Παράδειγμα: *Οι πελάτες που έχουν καταθέσεις και έχουν πάρει δάνειο.*

```
select customer–name
from borrower
where exists (select *
              from depositor
              where depositor.customer–name =
                   borrower.customer–name)
```

Φωλιασμένες Υποερωτήσεις (συνέχεια)

Ο τελεστής **not exists** μπορεί να χρησιμοποιηθεί για έλεγχο αν "η σχέση A περιέχει τη σχέση B": "**not exists** (B **except** A)"

Παράδειγμα: *Οι πελάτες που έχουν καταθέσεις σε όλα τα υποκαταστήματα του 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*)

Φωλιασμένες Υποερωτήσεις (συνέχεια)

4. Έλεγχος για Διπλές Εμφανίσεις

Ο τελεστής unique: επιστρέφει **true** ανν η υποερώτηση δεν έχει πολλαπλές όμοιες πλειάδες

Παράδειγμα: *Οι πελάτες που έχουν μόνο μια κατάθεση στο παράρτημα Perryridge.*

> **select** *T.customer–name*
> **from** *depositor* **as** *T*
> **where unique**
>   (**select** *R.customer–name*
>   **from** acount, depositor **as** *R*
>   **where** T.customer–name = R.customer–name **and**
>   R.acount–number = account.account–number **and**
>   acount.branch–name = *"Perryridge")*

Ο τελεστής nonunique

Παράδειγμα: Οι πελάτες που έχουν τουλάχιστον δυο καταθέσεις στο παράρτημα Perryridge.

Παραγόμενες Σχέσεις

Η SQL-92 δίνει τη δυνατότητα χρησιμοποιώντας το **as** να δοθεί ένα προσωρινό όνομα σε μία σχέση που προκύπτει από μια υποερώτηση.

Παράδειγμα: *Το μέσο υπόλοιπο για όλα τα υποκαταστήματα για τα οποία το μέσο υπόλοιπο είναι μεγαλύτερο των $1200*

select *branch–name*, *avg–balance*
from (**select** *branch–name*, **avg**(*balance*)
  from *depositor*
  group by *branch name*)
  as *result* (*branch name*, *avg–balance)*
where avg–balance > *1200*

Ορισμός Όψεων

Μπορούμε να ορίσουμε μια όψη χρησιμοποιώντας την εντολή:

create view *όνομα–όψης* **as** *ερώτηση*

Επίσης, μπορούν να προσδιοριστούν τα ονόματα των γνωρισμάτων άμεσα:

create view *όνομα–όψης*(*λίστα ονομάτων-γνωρισμάτων*) **as** *ερώτηση*

Παράδειγμα: *Μια όψη που περιλαμβάνει τα ονόματα όλων των υποκαταστημάτων και το άθροισμα των δανείων που έχουν γίνει από αυτά*

create view *branch-total-loan*(*branch–name*, *total–loan*) **as**
select *branch–name*, **sum**(*amount*)
from *loan*
group by *branch–name*

Τα ονόματα όψεων μπορεί να χρησιμοποιηθούν οπουδήποτε μπορεί να χρησιμοποιηθεί το όνομα μιας σχέσης.

Ο ορισμός της όψης παραμένει στην βάση δεδομένων, εκτός αν σβηστεί:

drop view *όνομα-όψης*

Τροποποιήσεις

1. Διαγραφή

Μπορούμε να σβήσουμε μόνο ολόκληρες πλειάδες και όχι συγκεκριμένα γνωρίσματα.

delete from R
where P

Σβήνει όλες τις πλειάδες της R για τις οποίες ισχύει το P.

Όταν λείπει το **where** σβήνονται όλες οι πλειάδες μιας σχέσης.

Παραδείγματα

(1) *Όλους τους λογαριασμούς του Smith*

delete from *depositor*
where *customer–name* = "Smith"

(2) *Όλους τους λογαριασμούς στα υποκαταστήματα που βρίσκονται στο Brooklyn.*

delete from *account*
where *branch–name* **in** (**select** *branch–name*
from *branch*
where *branch–city* = "Brooklyn")

Τροποποιήσεις (συνέχεια)

Αν και μπορούμε να σβήσουμε πλειάδες μόνο από μία σχέση τη φορά μπορούμε να αναφερθούμε σε περισσότερες από μια σχέσεις στην υποερώτηση του **where**

(3) *Όλους τους λογαριασμούς με υπόλοιπο μικρότερο από το μέσο υπόλοιπο στην τράπεζα.*

> **delete from** *account*
> **where** *balance* < (**select avg**(*balance*)
> **from** *account*)

Πρώτα γίνεται ο έλεγχος σε όλες τις πλειάδες και μετά αυτές διαγράφονται.

2. Εισαγωγή

Για να εισάγουμε δεδομένα σε μια σχέση είτε (α) προσδιορίζουμε την πλειάδα είτε (β) γράφουμε μια ερώτηση που το αποτέλεσμα της εισάγεται στη σχέση.

Παράδειγμα για το (α)

> **insert into** *account*
> **values** ("Perryridge", "A–9732", 1200)

Όταν με οποιαδήποτε σειρά, π.χ.,:

> **insert into** *account*(*account–number*,*branch–name*,*balance*)
> **values** ("A–9732","Perryridge", 1200)

Τροποποιήσεις (συνέχεια)

Παράδειγμα για το (β): *Για κάθε πελάτη που έχει πάρει δάνειο απο υποκατάστημα του Perryridge προστίθεται ως δώρο ένας λογαριασμός των $200*

> **insert into** *account*
>   **select** *branch–name*, *loan–number*, 200
>   **from** *loan*
>   **where** *branch–name* = "Perryridge"

Πρέπει πρώτα να υπολογιστεί το **select** πλήρως και μετά να γίνει η εισαγωγή. Τι αποτέλεσμα έχει η παρακάτω εντολή αν αυτό δε συμβαίνει;

> **insert into** *account*
>   **select** *
>   **from** *account*

Επίσης, εισαγωγή null τιμών:

> **insert into** *account*
>   **values** (*null* "A–9732", 1200)

Τροποποιήσεις (συνέχεια)

3, Ενημερώσεις

Παράδειγμα: *Αύξηση όλων των υπολοίπων που είναι μεγαλύτερα των $100 κατά 5% λόγω τοκισμού*

> **update** *account*
> **set** $balance = balance * 1.05$
> **where** $balance > 100$

Παράδειγμα: *Αύξηση όλων των υπολοίπων που είναι μεγαλύτερα από τον μέσο όρο κατά 5% λόγω τοκισμού*

> **update** *account*
> **set** $balance = balance * 1.05$
> **where** *balance* > **select avg**(*balance*)
> **from** *account*

Συνενώσεις Συνόλων

Η SQL–92 υποστηρίζει διάφορους τύπους συνενώσεων που συνήθως χρησιμοποιούνται στο **for**, αλλά μπορούν να χρησιμοποιηθούν οπουδήποτε μπορεί να χρησιμοποιηθεί μια σχέση.

Γενική σύνταξη:

$\textit{όνομα–σχέσης}_1$ $<$*τύπος–συνένωσης*$>$ $\textit{όνομα–σχέσης}_2$
$<$*συνθήκη–συνένωσης*$>$

ή

$\textit{όνομα–σχέσης}_1$ **natural** $<$*τύπος–συνένωσης*$>$ $\textit{όνομα–σχέσης}_2$

Τύποι Συνένωσης:

- **inner join**: εσωτερική (θήτα) συνένωση
- **left outer join**: αριστερή εξωτερική συνένωση
- **right outer join**
- **full outer join**

Συνθήκες Συνένωσης:

- **on** P, όπου P συνθήκη
- **using**$(A_1, A_2, \ldots, A_n)$: όπως η φυσική συνένωση, μόνο που τα γνωρίσματα που πρέπει να ταιριάζουν στη συνένωση είναι τα A_i. Τα A_i πρέπει να είναι γνωρίσματα κοινά και στις δυο σχέσεις και εμφανίζονται στο αποτέλεσμα μόνο μια φορά.

Συνενώσεις Συνόλων (συνέχεια)

Για την εσωτερική συνένωση η συνθήκη δεν είναι υποχρεωτική, όταν λείπει ισοδυναμεί με καρτεσιανό γινόμενο.

Οι λέξεις κλειδιά **inner** και **outer** είναι προαιρετικές.

natural: φυσική συνένωση, τα γνωρίσματα εμφανίζονται στο αποτέλεσμα με την εξής διάταξη: πρώτα αυτά με τα οποία έγινε η συνένωση (δηλ., αυτά που είναι κοινά και στις δύο σχέσεις), μετά τα υπόλοιπα της πρώτης σχέσης, και τέλος τα υπόλοιπα της δεύτερης σχέσης.

Παράδειγμα: *Τα ονόματα των πελατών που είτε έχουν καταθέσεις είτε έχουν πάρει δάνεια (αλλά όχι και τα δυο)*

> **select** *customer–name*
> **from** (*depositor* **natural full outer join** *borrower*)
> **where** *account–number* **is** *null* **or**
>   *loan–number* **is** *null*

Γλώσσα Ορισμού Δεδομένων (ΓΟΔ)

Σχετικά με το λογικό σχήμα, η ΓΟΔ SQL υποστηρίζει τους ορισμούς:

- του σχήματος κάθε σχέσης
- του πεδίου τιμών κάθε γνωρίσματος
- των περιορισμών ακεραιότητας

Τύποι Πεδίου Ορισμού

Για τον ορισμό του πεδίου ορισμού, οι διαθέσιμοι built-in τύποι περιλαμβάνουν:

- char(n) (σταθερού μήκους)
- varchar(n)
- int
- smallint
- numeric(p,d) (d από τα p ψηφία είναι στα δεξιά της υποδιαστολής)
- real, double precision
- float(n)
- date (ημερομηνία)
- time (ώρα)

Η SQL ως ΓΟΔ (συνέχεια)

Ο ορισμός πεδίου μπορεί να περιέχει τον προσδιοριμό **not null**

Επίσης, επιτρέπεται δημιουργία πεδίου:

> **create domain** person-name **char**(20)

Ορισμός Σχήματος

> **create table** $R(A_1 D_1, A_2 D_2, \ldots, A_n D_n)$,
> $<$*περιορισμοί-ορθότητας*$_1 >$,
> $\ldots$,
> $<$*περιορισμοί-ορθότητας*$_k >$

όπου R είναι το όνομα της σχέσης, A_i τα ονόματα των γνωρισμάτων, και D_i οι τύποι των αντίστοιχων πεδίων τιμών.

Επιτρεπτοί περιορισμοί ορθότητας είναι της μορφής:

> **primary key** $(A_{j_1}, A_{j_2}, \ldots A_{j_m})$
> **unique** $(A_{j_1}, A_{j_2}, \ldots A_{j_m})$
> **check** P

Παραδείγματα

(1)

> **create table** *customer*
> (*customer–name* **char**(20) **not null**,
> *customer–street* **char**(30),
> *customer–city* **char**(30),
> **primary key** (*customer–name*))

H SQL ως ΓΟΔ (συνέχεια)

(2)

create table *account*
(*account–number* **char**(10) **not null**,
branch–name **char**(15),
balance **integer**,
primary key (*account–number*)
check (*balance* >= 0))

Επίσης, πιο περίπλοκες συνθήκες, π.χ., για ξένα κλειδιά:

check (*branch–name* **in** (**select** *branch–name* **from** *branch*)

Μια καινούργια σχέση είναι αρχικά άδεια.

Για να σβηστεί ένα σχήμα:

drop table R

Διαφορά από

delete from R

Προσθήκη νέου γνωρίσματος:

alter table R **add** A D

προσθήκη σε μια σχέση R που ήδη υπάρχει του γνωρίσματος A με πεδίο D, η τιμή των πλειάδων της R στο καινούργιο γνώρισμα είναι *null*.

Διαγραφή γνωρίσματος:

alter table R **drop** A

Η SQL ως ΓΟΔ (συνέχεια)

Περιορισμοί Ακεραιότητας

1. Πεδίου ορισμού

Χρησιμοποιώντας την εντολή **check**:

Παραδείγματα

(1) Ελάχιστος ωρομίσθιο

create domain *hourly–wage* **numeric**(5, 2)
constraint *wage–value–test* **check**(**value** >= 4.00)

(2) Να μην περιέχει την τιμή null

create domain *account–number* **char**(10)
constraint *account–number–null–test* **check**(**value not null**)

(3) Να παίρνει συγκεκριμένες τιμές

create domain *account–type* **char**(10)
constraint *account–type–test*
check(**value in** ("Checking", "Saving"))

Η SQL ως ΓΟΔ (συνέχεια)

2. Περιορισμοί Αναφοράς

Σύνταξη:

foreign key (A_i) **references** A_j

Όταν μια πράξη παραβιάζει έναν περιορισμό αναφοράς απορρίπτεται εκτός αν έχει οριστεί:

- **on delete cascade**
- **on update cascade**

Παράδειγμα

create table

...

foreign key (*branch–name*) **references** *branch*

on delete cascade

on update cascade

...)

Ενσωματωμένη SQL

Προσπέλαση μιας ΒΔ μέσω μια γλώσσας προγραμματισμού γενικού σκοπού απαιτείται τουλάχιστον γιατί:

1. υπάρχουν ερωτήσεις που δε μπορούν να διατυπωθούν σε SQL, γιατί η SQL δεν έχει όλες τις δυνατότητες μιας γλώσσας προγραμματισμού γενικού σκοπού
2. μη-δηλωτικές εντολές (π.χ., εκτύπωση, επικοινωνία με το χρήστη) δε μπορούν να γίνουν μέσω της SQL

Ενσωμάτωση της SQL σε μια γλώσσα που καλείται *φιλόξενη* (host)

Σε αυτήν την περίπτωση, η επεξεργασία των ερωτήσεων γίνεται από τη ΒΔ, και το αποτέλεσμα γίνεται διαθέσιμο στο πρόγραμμα μια πλειάδα τη φορά

Ένας ειδικός προ-επεξεργαστής (preprocessor) αντικαθιστά τον ενσωματωμένο κώδικα της SQL με δηλώσεις και κλήσεις συναρτήσεων στη host γλώσσα και μετά μεταφράζεται το πρόγραμμα

Σύνταξη της μορφής:

EXEC SQL < embedded SQL statement > END-EXEC

Η ακριβής σύνταξη εξαρτάται από τη host γλώσσα

Ενσωματωμένη SQL (συνέχεια)

Χρησιμοποιούμε την εντολή: SQL INCLUDE, για να δηλώσουμε στον προ-επεξεργαστή που πρέπει να εισάγει τις δηλώσεις των μεταβλητών της SQL

Μεταβλητές της γλώσσας μπορεί να χρησιμοποιηθούν στην εντολή της SQL αν το σύμβολο : προηγείται του ονόματός τους

Για να γράψουμε μια ερώτηση σε SQL:

EXEC SQL
declare *c* **cursor for**
select *customer–name*, *customer–city*
from *deposit*, *customer*
where *deposit.customer–name* = *customer.customer–name*
and *deposit.balance* > :*amount*
END-EXEC

Η παραπάνω εντολή δεν προκαλεί την εκτέλεση της ερώτησης, για να εκτελεστεί:

EXEC SQL **open** *c* END-EXEC

Το αποτέλεσμα σώζεται σε μια προσωρινή σχέση, αν υπάρχει λάθος το διαγνωστικό μήνυμα σώζεται σε μια ειδική μεταβλητή (SQLCA)

Ενσωματωμένη SQL (συνέχεια)

Οι πλειάδες του αποτελέσματος γίνονται διαθέσιμες στο πρόγραμμα μέσω μιας σειράς από **fetch** εντολές, χρειάζεται μια μεταβλητή της host γλώσσας για κάθε γνώρισμα

EXEQ SQL **fetch** *c* **into** :*cn*, :*cc* END-EXEC

Για να σβήσουμε τη προσωρινή σχέση:

EXEC SQL **close** *c* END-EXEC

Επίσης:

EXEC SQL < any valid **update, insert** or **delete** > END-EXEC

Δυναμική SQL

Τα προγράμματα μπορούν να δημιουργούν ερωτήσεις σε SQL ως συμβολοσειρές δυναμικά κατά την εκτέλεση και είτε να τα εκτελούν αμέσως είτε να τα "προετοιμάζουν" (τα μεταφράζουν) για να χρησιμοποιηθούν αργότερα

Παράδειγμα:

```
char * sqlprog = "update acount
   set balance = balance * 1.05
   where acount–number = ?"
EXEC SQL prepare dynprog from :sqlprog;
char account[10] = "A-101";
EXEC SQL execute dynprog using :account;
```

Άλλα Χαρακτηριστικά

- Γλώσσες 4ης Γενιάς
- Έννοια του session μεταξύ ενός client και του server του ΣΒΔ
- Δημιουργία σχήματος
  create schema και
  Σβήσιμο σχήματος
  drop schema