Η Γλώσσα SQL

Μέρος α’

SQL - ΕΙΣΑΓΩΓΗ

- ***SQL (Structured Query Language)*** είναι η τυποποιημένη "standard" γλώσσα στις Σχεσιακές Βάσεις. Η πρώτη χρήση ήταν στο πρότυπο σύστημα της IBM, που ονομάστηκε SYSTEM-R, το οποίο αναπτύχθηκε στα ερευνητικά εργαστήρια της εταιρείας (San Jose, California) στα μέσα της δεκαετίας το 1970. Η SQL έχει υποστεί πολλές τροποποιήσεις.
- **Υπάρχουν 4 βασικές εντολές:**
  - **select** (δεν είναι το ίδιο με το SELECTION στην Άλγεβρα)
  - **insert**
  - **update**
  - **delete**
- **Το αποτέλεσμα μιας εντολής / πράξης σε Σχέσεις είναι (πάντα) μια νέα Σχέση**

SQL - Διαχρονικά

- Η γλώσσα SQL - παγκόσμια σταθερά σε ΒΔ
  - Sequel 75 (System R)
  - SQL86
  - SQL89
  - SQL92
  - SQL3
  - SQL:2003
  - SQL:2006
  - SQL:2008
  - SQL:2011

Βασική δομή

- Η SQL βασίζεται σε σχεσιακές πράξεις
- Ένα τυπικό SQL query είναι της μορφής:

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

  - A_is αντιπροσωπεύει γνωρίσματα
  - r_is αντιπροσωπεύει σχέσεις
  - P είναι ένα κατηγόρημα
- Το query είναι ισοδύναμο με την έκφραση της σχεσιακής άλγεβρας

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

- Το αποτέλεσμα του SQL query είναι μια σχέση

The select Clause

- Η **select** clause είναι η λίστα των αναγνωριστικών που θέλουμε να υπάρχουν στα αποτελέσματα του query
  - Αντιστοιχεί σε πράξη προβολής (projection) της σχεσιακής άλγεβρας
- Βρες τα ονόματα όλων των branches της σχέσης *loan*

  select *branch-name*
  from *loan*
- Στη σχεσιακή άλγεβρα το query θα ήταν:

  $\Pi_{\text{branch-name}}(loan)$
- NOTE: η SQL δεν επιτρέπει τον χαρακτήρα '-' στα ονόματα,
  - » Μπορεί να αντικατασταθεί με π.χ. *branch_name*

The select Clause (2)

- Η SQL επιτρέπει duplicates στις σχέσεις και στα αποτελέσματα των queries
- Για να μην υπάρχουν duplicates χρησιμοποιούμε τη λέξη-κλειδί **distinct** μετά το **select.**
- Βρες τα ονόματα όλων των branches στη σχέση *loan* και αφαίρεσε τα διπλότυπα

  select distinct *branch-name*
  from *loan*

- Η λέξη-κλειδί **all** καθορίζει ότι τα duplicates δεν πρέπει να αφαιρεθούν

  select all *branch-name*
  from *loan*

The select Clause (3)

- Το * στο select clause σημαίνει “όλα τα γνωρίσματα”

  select *
  from *loan*

- Το **select** clause μπορεί να περιέχει αριθμητικές εκφράσεις που εμπλέκουν πράξεις (+, –, *, /) και δουν πάνω σε σταθερές ή γνωρίσματα πλειάδων
- Π.χ.:

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

The where Clause

- Το **where** clause καθορίζει τις συνθήκες που πρέπει να ικανοποιεί το αποτέλεσμα
  - Αντιστοιχεί στο selection predicate της σχεσιακής άλγεβρας
- Βρες όλους τους κωδικούς δανείων στο Perryridge branch με ποσά μεγαλύτερα του $1200.

  select *loan-number*
  from *loan*
  where *branch-name* = ‘Perryridge’ **and** *amount* > 1200
- Λογικές πράξεις **and, or,** and **not.**

The where Clause (2)

- Η SQL έχει και operator σύγκρισης **between**
- Π.χ. Βρες τους κωδικούς δανείων με ποσά ανάμεσα σε $90,000 και $100,000

select *loan-number*
from *loan*
where *amount* **between** 90000 **and** 100000

The from Clause

- Το **from** clause περιέχει όλες τις σχέσεις που εμπλέκονται στο query
  - Αντιστοιχεί στο καρτεσιανό γινόμενο της σχεσιακής άλγεβρας
- Βρες το καρτεσιανό γινόμενο *borrower x loan*

  select *
  from *borrower*, *loan*

- Βρες το όνομα, κωδικό δανείου και ποσό για όλους τους πελάτες που έχουν δάνειο στο υποκατάστημα Perryridge

loan (loan-number, branch-name, amount)
borrower (customer-name, loan-number)

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

The Rename Operation

- Η SQL επιτρέπει τη μετονομασία σχέσεων και γνωρισμάτων χρησιμοποιώντας το **s** clause:

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

- Βρες το όνομα, κωδικό δανείου και ποσό όλων των πελατών. Μετονόμασε το *loan-number* σε *loan-id*.

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

SQL: Χρήση Μεταβλητών Πλειάδας

- **Τα ονόματα των Σχέσεων μπορεί να χρησιμοποιηθούν αντί για τις μεταβλητές πλειάδος**
- Αν το Σύστημα μπορεί να καταλάβει σε ποια Σχέση κάθε Γνώρισμα ανήκει, τότε οι μεταβλητές πλειάδος μπορεί να είναι ***συνεπαγόμενες (implicit)***

Βρες το όνομα, κωδικό δανείου και ποσό για όλους τους πελάτες που έχουν δάνειο στο υποκατάστημα Perryridge

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

Tuple Variables

- Οι μεταβλητές πλειάδων ορίζονται στο **from** clause με τη χρήση του **as** clause.
- Βρες τα ονόματα όλων των πελατών με δάνεια και τον κωδικό των δανείων τους.

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

SQL: Παραδείγματα στην ΒΔ Ναυτικών (1)

- Επανερχόμαστε στην ΒΔ των Ναυτικών:

  SAILORS (Sid, SName, Rating)

  BOATS (Bid, BName, Color)

  RESERVE (Sid, Bid, Date)

- SQUERY1: *Βρες τα ονόματα των ΝΑΥΤΙΚΩΝ που έχουν κάνει κράτηση για το ΣΚΑΦΟΣ με νούμερο 2*

```
select    SName
from      SAILORS,  RESERVE
where     SAILORS.Sid = RESERVE.Sid  and
          RESERVE.Bid = 2
```

SQL: Παραδείγματα στην ΒΔ Ναυτικών (2)

- SQUERY2: *Βρες τα ονόματα των ΝΑΥΤΙΚΩΝ που έχουν κρατήσει ΣΚΑΦΗ με χρώμα κόκκινο*

```
select    SName
from      SAILORS,  BOATS, RESERVE
where     SAILORS.Sid = RESERVE.Sid and
          RESERVE.Bid = BOATS.Bid and
          BOATS.Color = "red"
```

- SQUERY3: *Βρες τα χρώματα των σκαφών που έχει κρατήσει η ναυτικός με το όνομα Ελένη*

```
select    b.Color
from      SAILORS as s,  BOATS as b, RESERVE as r
where     s.Sid = r.Sid and r.Bid =b.Bid and s.SName= "eleni"
```

SQL: Παραδείγματα στην ΒΔ Ναυτικών (3)

- SQUERY4: *Βρες τα ονόματα των ΝΑΥΤΙΚΩΝ που έχουν κρατήσει τουλάχιστον ένα ΣΚΑΦΟΣ*

```
select    s.SName
from      SAILORS as s,  RESERVE as r
where     s.Sid = r.Sid
```

- SQUERY5: *Βρες τα ονόματα των ΝΑΥΤΙΚΩΝ που έχουν κρατήσει ένα κόκκινο* ***ή*** *ένα πράσινο ΣΚΑΦΟΣ*

```
select    s.SName
from      SAILORS as s,  BOATS as b, RESERVE as r
where     s.Sid = r.Sid and r.Bid = b.Bid and
          (b.Color = "red"  or  b.Color = "green")
```

SQL: Παραδείγματα στην ΒΔ Ναυτικών (4)

- **SQUERY6**: *Βρες τα ονόματα των ΝΑΥΤΙΚΩΝ που έχουν κρατήσει ένα κόκκινο* ***και*** *ένα πράσινο ΣΚΑΦΟΣ*

```
select   s.SName
from     SAILORS as s,  BOATS as b1,  RESERVE as r1,
         BOATS as b2,  RESERVE as r2
where    s.Sid = r1.Sid  and  r1.Bid =  b1.Bid  and
         b1.Color = "red"   and
         s.Sid = r2.Sid  and  r2.Bid = b2.Bid  and
         b2.Color = "green"
```

Πράξεις με συμβολοσειρές

- Η SQL χρησιμοποιεί ειδικά σύμβολα για string-matching (ταίριασμα συμβολοσειρών) :
  - percent (%). Ο % χαρακτήρας ταιριάζει με οποιαδήποτε συμβολοσειρά.
  - underscore (_). Ο _ χαρακτήρας ταιριάζει με οποιοδήποτε χαρακτήρα.

- Find the names of all customers whose street includes the substring "Main".

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

- Match the name "Main%"

  like 'Main\%' **escape** '\'

- Η SQL υποστηρίζει πολλές πράξεις σε συμβολοσειρές
  - concatenation (με το "||")
  - μετατροπή από κεφαλαία σε πεζά και αντίστροφα
  - μήκος συμβολοσειράς, κλπ.

Ταξινόμηση των αποτελεσμάτων

- Παρουσίασε με αλφαβητική σειρά τα ονόματα όλων των πελατών που έχουν δάνειο στο υποκατάστημα Perryridge

  select distinct *customer-name*
  from *borrower*, *loan*
  where *borrower loan-number* = *loan.loan-number* **and**
  branch-name = ‘Perryridge’
  order by *customer-name*

- Καθορίζουμε **desc** για φθίνουσα σειρά και **asc** για αύξουσα, για κάθε attribute. Η αύξουσα είναι default.
  - Π.χ., **order by** *customer-name* **desc**

Πράξεις συνόλων

- Οι πράξεις συνόλων **union, intersect,** και **except** δρουν πάνω σε σχέσεις και αντιστοιχούν στις πράξεις της σχεσιακής αλγεβρας $\cup$, $\cap$, $-$.
- Οι πράξεις συνόλων αφαιρούν αυτόματα τα duplicates; Για να διατηρηθούν τα duplicates χρησιμοποιείται το **union all, intersect all** and **except all.**

  Αν ένα tuple εμφανίζεται *m φορές στο r και n φορές στο s*, τότε:

  - *m* + *n* times in *r* **union all** *s*
  - min(*m*,*n)* times in *r* **intersect all** *s*
  - max(0, *m* – *n)* times in *r* **except all** *s*

Πράξεις συνόλων

- Find all customers who have a loan, an account, or both:

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

- Find all customers who have both a loan and an account.

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

- Find all customers who have an account but no loan.

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

Συναθροιστικές συναρτήσεις

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

SQL Συναθροιστικές Συναρτήσεις

- Η SQL υποστηρίζει **ΠΕΝΤΕ Συναθροιστικές Συναρτήσεις (AGGREGATE FUNCTIONS)**

  - **count ([DISTINCT] X)** *:αριθμός (διακριτών) τιμών στο X*
  - **sum ([DISTINCT] X)** *: άθροισμα των (διακριτών) τιμών στο X*
  - **avg ([DISTINCT] X)** *: μέση τιμή των (διακριτών) τιμών στο X*
  - **max (X)** *:μέγιστη τιμή στο X*
  - **min (X)** *:ελάχιστη τιμή στο X*

- Οι **Συναθροιστικές Συναρτήσεις επιστρέφουν ΜΙΑ τιμή**

Συναθροιστικές Συναρτήσεις (2)

- Βρες τον αριθμό των tuples στη σχέση customer

select count (*)
from *customer*

- Βρες το αριθμό των καταθετών της τράπεζας

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

Συναθροιστικές Συναρτήσεις (3)

- Το σύνολο στο οποίο οι **Συναθροιστικές Συναρτήσεις** εφαρμόζονται μπορεί να περιοριστεί από την **where**-πρόταση
- Find the average account balance at the Perryridge branch.

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

Ομαδοποίηση– Group By

- Συσσωρεύσεις ή ομάδες πλειάδων υπολογίζονται με την **GROUP BY** clause
- Ομαδοποίησε τους λογαριασμούς ανά υποκατάστημα

select *branch-name, account-number*
from *account*
group by *branch-name*

Σημειώνεται ότι το Γνώρισμα ομαδοποίησης πρέπει να παρουσιάζεται στην **select** πρόταση

Ομαδοποίηση και συναθροιστικές συναρτήσεις

- Βρες τον αριθμό των καταθετών ανά υποκατάστημα

select *branch-name*, **count (distinct** *customer-name)*
from *depositor, account*
where *depositor.account-number = account.account-number*
group by *branch-name*

Οι ομαδοποιήσεις και συναρτήσεις εφαρμόζονται ***META*** τη Συνένωση των Σχέσεων

Ομαδοποίηση – Having Clause

- Η έκφραση που πρέπει να ικανοποιείται ***για κάθε*** ομάδα που συγκροτείται στην **group by-** clause, μπαίνει σε μια **having** clause
- Βρες τα ονόματα των υποκαταστημάτων όπου το μέσο υπόλοιπο λογαριασμού είναι μεγαλύτερο από $1,200.

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

Note: Τα predicates στο **having** εφαρμόζονται μετά τη συγκρότηση των ομάδων, ενώ τα predicates του where clause πριν

SQL -- Πλήρης Μορφή

select [**distinct**] *target_list*
from *tuple_variable_list*
[**where** *qualification*]
[**group by** *grouping_attributes*]
[**having** *group_condition*]
[**order by** *target_list_subset*]

- Μια ερωταπόκριση **αποτιμάται** ***πρώτα*** με την εφαρμογή της WHERE πρότασης, ***μετά*** των GROUP-BY και HAVING (προαιρετικά), και ***τέλος*** της SELECT πρότασης (στόχος λίστα γνωρισμάτων) – *με ταξινόμηση* των πλειάδων στην απάντηση ή διαγραφή διπλών τιμών, αν απαιτείται από την ORDER BY πρόταση ή το DISTINCT (επίσης προαιρετικά).

Null Values

- Άγνωστη τιμή ή τιμή που δεν υπάρχει/δεν την ξέρουμε
- Το predicate **is null** μπορεί να χρησιμοποιηθεί για έλεγχο null τιμών
  - E.g. Βρες όλους τους κωδικούς δανείων στη σχέση loan με null τιμή στο γνώρισμα amount.

    select loan-number
    from loan
    where amount **is null**
- Το αποτέλεσμα μιας αριθμητικής έκφρασης με null είναι null
  - Π.χ., 5 + null επιστρέφει null
- Οι συναθροιστικές συναρτήσεις αγνοούν τις τιμές null

Null και η λογική των 3 τιμών

- Κάθε σύγκριση με *null* επιστρέφει *unknown*
  - *E.g. 5 < null or null <> null or null = null*
- Λογικές πράξεις με *unknown*:
  - OR: (*unknown* **or** *true*) = *true*

    (*unknown* **or** *false*) = *unknown*

    (*unknown* **or** *unknown*) = *unknown*
  - AND: *(true* **and** *unknown) = unknown*,

    (false **and** *unknown) = false*,

    (unknown **and** *unknown) = unknown*
  - NOT: *(***not** *unknown) = unknown*
  - "*P* **is unknown"** είναι true αν το *P* είναι *unknown*
- Το αποτέλεσμα ενός **where** clause predicate που είναι unknown, το διαχειριζόμαστε ως *false*

Null τιμές και συναθροιστικές συναρτήσεις

- Το σύνολο όλων των ποσών των δανείων

  select sum (*amount)*
  from *loan*

  – Αγνοεί τα null amounts
  – Αποτέλεσμα null αν δεν υπάρχει κανένα non-null amount
  – Όλες οι συναθροιστικές συναρτήσεις εκτός της **count(*)** αγνοούν τις τιμές null στο γνώρισμα της συνάθροισης

Εμφωλιασμός

- **ΕΜΦΩΛΙΑΣΜΟΣ (nesting) SQL Ερωταποκρίσεων**: Ένα πλήρες SELECT query (αποκαλείται ***nested*** query) μπορεί να υπάρχει μέσα στην πρόταση WHERE ενός άλλου query (αποκαλείται, το ***outer*** query)
- Συνήθως χρησιμοποιείται για έλεγχο συμμετοχής σε σύνολο και σύγκριση συνόλων

Example Query

- Find all customers who have both an account and a loan at the bank.

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

Κάθε ερωταπόκριση που χρησιμοποιεί τον τελεστή **IN** (ελέγχει τη συμμετοχή σε σύνολο) μπορεί πάντα να εκφραστεί χωρίς εμφωλιασμό (flat query)

```
select distinct customer-name
        from borrower, depositor
        where borrower.customer-name=depositor.customer-name
```

Εμφωλιασμός

- Παρόμοιοι τελεστές με το **IN** *είναι*:
  - **not in** *(ελέγχει τη μη συμμετοχή σε σύνολο)*
  - *OP* **any** *(OP συσχέτιση με κάποια πλειάδα σε σύνολο)*
  - *OP* **all** *(OP συσχέτιση με όλες τις πλειάδες σε σύνολο)*

  where ***OP*** $\in \{ =, \neq, <, >, \leq, \geq \}$

- Find all customers who have a loan at the bank but do not have an account at the bank

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

Εμφωλιασμός – σύγκριση

- Η SQL επίσης παρέχει τελεστές για σύγκριση συνόλων:
  - **contains, not contains** *(υπερσύνολο ή όχι)*
  - **exists, not exists** *(κενό σύνολο ή όχι)*
- CQUERY11: *Παρουσίασε τους Υπαλλήλους που εργάζονται σε όλα τα έργα που ελέγχονται από το Τμήμα 4*

```
select      Name
from        EMPLOYEE e
where       (select  w.PNumber
             from    WORKS_ON
             where   w.SSN = e.SSN)
      contains
            (select PNumber
             from    PROJECT
             where   DNumber = 4)
```

Έλεγχος για κενές σχέσεις

- Το **exists** επιστρέφει **true** αν το υπο-query δεν επιστρέφει κενό σύνολο
- **exists** $r \Leftrightarrow r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$

Example Query

- Find all customers who have an account at all branches located in 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))
```

Έλεγχος για απουσία διπλότυπων

- Το **unique** ελέγχει αν ένα υπο-query έχει διπλότυπα αποτελέσματα
- Find all customers who have at most one account at the Perryridge branch.

select *T.customer-name*
from *depositor* **as** *T*
where unique (

select *R.customer-name*
from *account, depositor* **as** *R*
where *T.customer-name* = *R.customer-name* **and**
R.account-number = *account.account-number* **and**
account.branch-name = ‘Perryridge’)

Έλεγχος για απουσία διπλότυπων (2)

- Find all customers who have at least two accounts at the Perryridge branch.

```
select distinct T.customer-name
from depositor T
where not unique (
        select R.customer-name
        from account, depositor as R
        where T.customer-name = R.customer-name
and
        R.account-number = account.account-number
and
        account.branch-name = 'Perryridge')
```