SQL

Τι είδαμε μέχρι τώρα

Δύο γλώσσες ερωτήσεων που αποτελούν το θεωρητικό υπόβαθρο

Σχεσιακή άλγεβρα: μια άλγεβρα συνόλων που αφορά πράξεις πάνω σε σχέσεις

Σχεσιακό λογισμό (πλειάδων): δηλωτικό τρόπο έκφρασης ερωτήσεων

SQL

- *Ειδικού σκοπού γλώσσα προγραμματισμού για βάσεις δεδομένων*
- Η “standard” γλώσσα για σχεσιακές βάσεις δεδομένων.
- Δηλωτική (declarative) (αν και έχει κάποια στοιχεία διαδικαστικού προγραμματισμού)
- αρχικά Sequel (Structured English Query language) στην IBM ως μέρος του System R,
  - τώρα SQL (Stuctured Query Language)
- SQL-89, SQL-92, SQL-99, …

SQL

- DDL (Data Definition Language) Γλώσσα Ορισμού Δεδομένων (ΓΟΔ): ορισμός, δημιουργία, τροποποίηση και διαγραφή *σχήματος* – *την είδαμε σε προηγούμενο μάθημα*
- DML (Data Manipulation Language) Γλώσσα Χειρισμού Δεδομένων (ΓΟΔ)
  - εισαγωγή, τροποποίηση, διαγραφή δεδομένων - *την είδαμε σε προηγούμενο μάθημα*
  - επιλογή δεδομένων (γλώσσα ερωτήσεων, query language)

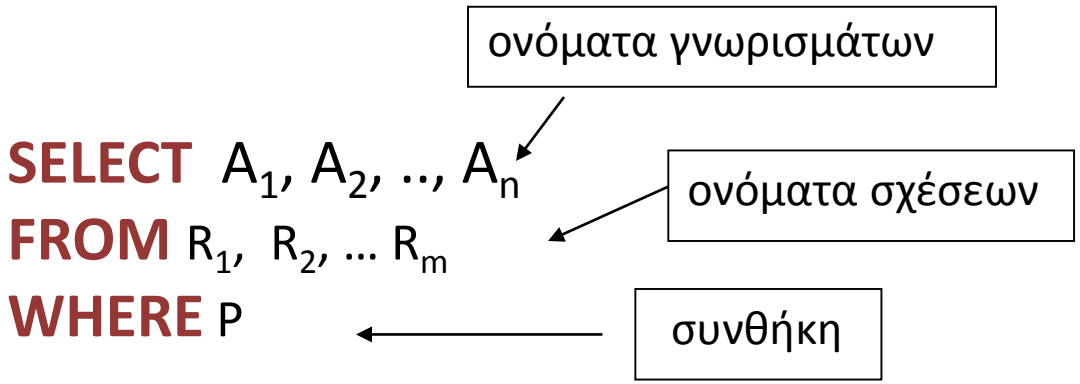
Προδιαγραφές ασφάλειας - χρήστες και δικαιώματα.

Διαφορές στην υποστήριξη της SQL σε διάφορα σχεσιακά ΣΔΒΔ (πχ Oracle SQL, MySQL, SQLite, κλπ)

Βασική Δομή Ερώτησης

Βασική Δομή

Η βασική δομή μιας ερώτησης σε SQL έχει την εξής μορφή:



Ισοδύναμο του: $\pi_{A_1, A_2, .., A_n} (\sigma_P (R_1 \times R_2 \times \ldots R_m))$

select

SELECT A1, A2, .., An
FROM R_1, R_2, … R_m
WHERE P

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

SELECT αντιστοιχεί στην πράξη της *προβολής* της σχεσιακής άλγεβρας

Ποια γνωρίσματα θέλουμε να υπάρχουν στο αποτέλεσμα της ερώτησης.

from

SELECT A1, A2, .., An
FROM R_1, R_2, … R_m
WHERE P

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

FROM αντιστοιχεί στην πράξη του *καρτεσιανού γινομένου* της σχεσιακής άλγεβρας.

Ποιες σχέσεις θα χρησιμοποιηθούν για τον υπολογισμό του αποτελέσματος.

where

SELECT $A_1, A_2, .., A_n$
FROM R1, R2, … Rm
WHERE P

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

WHERE αντιστοιχεί στη συνθήκη της πράξης της *επιλογής* στη σχεσιακή άλγεβρα.

Το κατηγόρημα P έχει γνωρίσματα των σχέσεων που εμφανίζονται στο FROM.

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Ονόματα ηθοποιών που παίζουν στην ταινία Gone by the Wind

SELECT Name
FROM Plays
WHERE Title = 'Gone by the Wind‘;

select

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

Παράδειγμα: Ονόματα όλων των ηθοποιών που έχουν παίξει σε ταινίες

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

SELECT Name
FROM Plays;

select distinct

ΠΡΟΣΟΧΗ: Δε γίνεται απαλοιφή των διπλών εμφανίσεων.

- 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 Name
FROM Plays;
```

Πόσες φορές εμφανίζεται το όνομα ενός ηθοποιού αν δεν υπάρχει το DISTINCT;

select *

Επιλογή όλων των γνωρισμάτων

SELECT *
FROM Plays;

Η «μικρότερη» SQL ερώτηση (μας δίνει το περιεχόμενο του αντίστοιχου πίνακα)

select

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

SELECT Title, Year, Duration/60, Type
FROM Movie;

Επιστρέφει μια σχέση ίδια με τη σχέση Ταινία μόνο που το γνώρισμα διάρκεια μας δίνει τις ώρες (έχει διαιρεθεί με το 60)

where

Συνθήκη του where

Λογικοί τελεστές: **and**, **or**, **not**

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

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Τον τίτλο όλων των ταινιών που γυρίστηκαν μετά το 1995 και είναι ασπρόμαυρες

```
SELECT DISTINCT Title
FROM Movie
WHERE Year > 1995 AND Type = 'Ασπρόμαυρη';
```

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Χρήση του between :

```
SELECT DISTINCT Title
FROM Movie
WHERE Year BETWEEN 1990 AND 1995;
```

αντί του

```
SELECT DISTINCT Title
FROM Movie
WHERE Year  >= 1990  AND Year <= 1995;
```

Βασική Δομή

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

  <όνομα-σχέσης>. <όνομα-γνωρίσματος>

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα *φυσικής συνένωσης*:
Τους ηθοποιούς (το όνομα τους) που γεννήθηκαν πριν το 1950 και έπαιξαν σε ταινίες μετά το 2010

```
SELECT DISTINCT Name
FROM  Plays, Actor
WHERE   Actor.Year-of-Birth < 1950 AND
        Play.Year > 2010 AND
        Actor.Name = Plays.Name;
```

Προσοχή στις συνθήκες συνένωσης

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

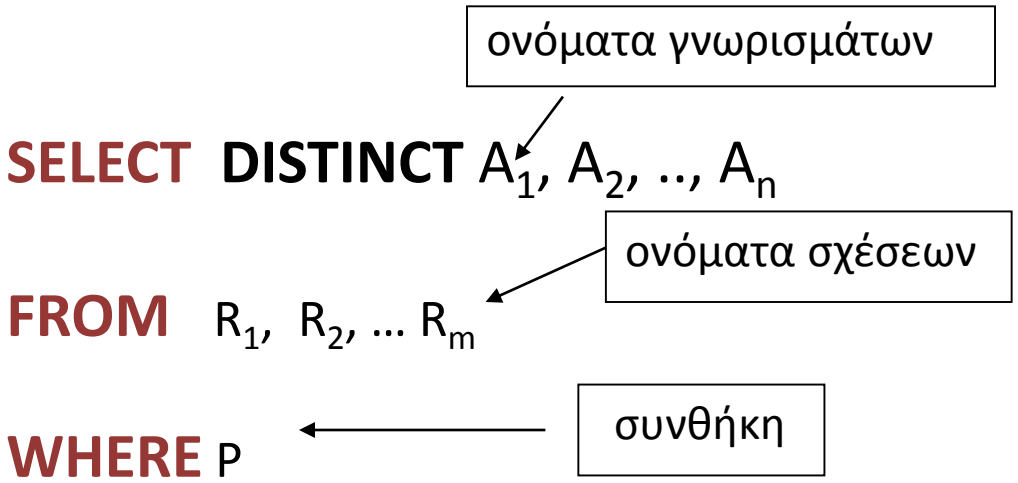
Παράδειγμα φυσικής συνένωσης:
Τους ηθοποιούς που παίζουν σε ασπρόμαυρες ταινίες

SELECT DISTINCT Name
FROM Plays, Movie
WHERE Type = 'Ασπρόμαυρη' **AND**
Plays.Title = Movie.Title **AND** Plays.Year = Movie.Year

Συνθήκη συνένωσης

Βασική Δομή (επανάληψη)

Η βασική δομή μιας ερώτησης σε SQL έχει την εξής μορφή:



Ισοδύναμο του: $\pi_{A_1, A_2, .., A_n} (\sigma_P (R_1 \times R_2 \times \ldots R_m))$

Βασική Δομή (επανάληψη)

Select

- ✓ Διαγραφή διπλότιμων: SELECT DISTINCT
- ✓ SELECT * (όλα τα γνωρίσματα)

Συνθήκη του where

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

Τα αποτελέσματα μιας ερώτησης ΔΕΝ αποθηκεύονται

Παραδείγματα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

1. Όλα τα συστατικά που αρέσουν σε φοιτητές
2. Τα συστατικά που αρέσουν στον φοιτητή Δημήτρη
3. Τα συστατικά της πίτσας Σπέσιαλ
4. Τις πίτσες που έχουν συστατικά που αρέσουν στον φοιτητή Δημήτρη

PIZZA

Name	Ingredient
Vegetarian	μανιτάρι
Vegetarian	ελιά
Χαβάη	ανανάς
Χαβάη	ζαμπόν
Σπέσιαλ	ζαμπόν
Σπέσιαλ	μπέικον
Σπέσιαλ	μανιτάρι
Ελληνική	ελιά

LIKES

Student	Ingredient
Δημήτρης	μανιτάρι
Κώστας	ζαμπόν
Μαρία	ελιά
Κατερίνα	μανιτάρι
Μαρία	ζαμπόν
Δημήτρης	μπέικον
Μαρία	ανανάς

SQL

Περισσότερα για τη γλώσσα ερωτήσεων

- Πράξεις με Συμβολοσειρές
- Διάταξη Πλειάδων
- Αλλαγή Ονόματος
- Μεταβλητές Πλειάδων
- Η τιμή null

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

Πράξεις με συμβολοσειρές

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα:

Οι τίτλοι όλων των ταινιών που περιέχουν τη λέξη Θάλασσα

SELECT DISTINCT Title
FROM Movies
WHERE Title **LIKE** '%Θάλασσα%';

Πολλές ακόμα πράξεις διαθέσιμες.

Διάταξη Πλειάδων

Χρήση του ORDER BY ώστε οι πλειάδες στο αποτέλεσμα να είναι ταξινομημένες με βάση το αντίστοιχο γνώρισμα

Default: αύξουσα διάταξη

```
SELECT DISTINCT Title,  Year
FROM Plays
WHERE Name = 'Robert De Niro'
ORDER BY Year;
```

Διάταξη Πλειάδων

Default: αύξουσα διάταξη
Αλλά και άμεσος προσδιορισμός χρησιμοποιώντας το ASC (αύξουσα) ή το DESC (φθίνουσα). Επίσης, ταξινόμηση με βάση πολλά γνωρίσματα.

Παράδειγμα:

SELECT *
FROM Movie
ORDER BY Year **DESC**, Title **ASC**;

Η ταξινόμηση είναι δαπανηρή λειτουργία.

Περιορισμός μεγέθους αποτελέσματος

Περιορισμό του μεγέθους του αποτελέσματος με χρήση του LIMIT <k>

Σε συνδυασμό ή όχι με το order by:

αν δεν υπάρχει το order by το LIMIT k μας δίνει κάποιες τυχαίες k πλειάδες από το αποτέλεσμα – αν υπάρχει το order by μας δίνει τις πρώτες k

```
SELECT Title, Year
FROM Plays
WHERE Name = 'Robert De Niro'
ORDER BY Year DESC
LIMIT 8;
```

8 από τις πιο πρόσφατες -- αν δεν υπάρχει το order by, δίνει 8 ***τυχαίες***

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

Αλλαγή Ονόματος

Για παράδειγμα:

SELECT Title, Year, Duration/60 **AS** Hourly-Duration, Type
FROM Movie;

Αλλαγή Ονόματος

Χρήσιμο όταν

(α) όταν έχουμε αριθμητικές εκφράσεις στο SELECT και δεν έχουν όνομα

(β) όταν θέλουμε να αλλάξουμε το όνομα του γνωρίσματος στο αποτέλεσμα

(γ) δυο σχέσεις του FROM έχουν γνωρίσματα με το ίδιο όνομα

Μεταβλητές πλειάδων

Μια *μεταβλητή πλειάδας* μπορεί να οριστεί στο FROM χρησιμοποιώντας το AS:

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

SELECT DISTINCT Name
FROM Plays **AS** Π, Movie **AS** T
WHERE Π.Title = T.Title **AND** Π.Year = T.Year **AND** Type = 'Ασπρόμαυρη';

Μεταβλητές πλειάδων

✓ Οι μεταβλητές πλειάδων είναι ιδιαίτερα χρήσιμες όταν θέλουμε να συγκρίνουμε δυο πλειάδες της ίδιας σχέσης (με συνένωση - self-join).

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Τα ονόματα όλων των ταινιών που έχουν διάρκεια μεγαλύτερη τουλάχιστον από μία ταινία που γυρίστηκε το 1995

SELECT DISTINCT T.Title
FROM Movie **AS** T, Movie **AS** S
WHERE T.Duration > S. Duration **AND** S.Year = 1995;

Βασική Δομή Ερώτησης

SELECT **[DISTINCT]** $A_1, A_2, .., A_n$
FROM $R_1, R_2, \ldots R_m$
WHERE P

Βασική Δομή Ερώτησης

SELECT **[DISTINCT]** $A_1, A_2, .., A_n$
FROM $R_1, R_2, \ldots R_m$
WHERE P
ORDER BY A_i
LIMIT k;

Παραδείγματα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

1. Όλα τα συστατικά που υπάρχουν σε πίτσες σε αλφαβητική σειρά
2. Τρία συστατικά που αρέσουν σε φοιτητές
3. Τα συστατικά της πίτσας Σπέσιαλ που αρέσουν στο Δημήτρη
4. Τα συστατικά που αρέσουν σε δύο τουλάχιστον φοιτητές

Η τιμή null

Χρήση της λέξης κλειδί IS NULL (IS NOT NULL) σε μια συνθήκη για να ελέγξουμε αν μια τιμή είναι null.

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
SELECT Name
FROM Actor
WHERE Address IS NULL;
```

Λογική Τριών Τιμών

Η SQL λογική τριών τιμών με τιμές TRUE, FALSE, και ΑΓΝΩΣΤΟ (null)

Στο αποτέλεσμα του SELECT-FROM-WHERE ανήκουν μόνο οι πλειάδες που ικανοποιούν την συνθήκη του where (η έκφραση έχει την τιμή TRUE)

NOT	
TRUE	FALSE
FALSE	TRUE
ΑΓΝΩΣΤΟ (NULL)	ΑΓΝΩΣΤΟ (NULL)

Λογική Τριών Τιμών

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKOWN	TRUE	UNKNOWN	UNKNOWN

P = Q, αν ένα από τα δύο είναι UNKNOWN δίνει UNKNOWN

Η τιμή null

Εμφάνιση null

- Σε αριθμητικές πράξεις: το αποτέλεσμα είναι null όταν οποιαδήποτε τιμή είναι null

- *Σε συναθροιστικές συναρτήσεις: αγνοείται πλην από το COUNT(*)*

Επανάληψη

Πράξεις με Συμβολοσειρές

Η πιο συνηθισμένη πράξη είναι ταίριασμα προτύπων:

% ταιριάζει οποιαδήποτε συμβολοσειρά

_ ταιριάζει οποιοδήποτε χαρακτήρα

Σύγκριση χρησιμοποιώντας το LIKE, NOT LIKE

Διάταξη των Πλειάδων

Χρήση του ORDER BY ώστε οι πλειάδες στο αποτέλεσμα να είναι ταξινομημένες με βάση το αντίστοιχο γνώρισμα

Default: αύξουσα διάταξη, αλλά και άμεσα χρησιμοποιώντας το ASC (αύξουσα) ή το DESC (φθίνουσα).

Επανάληψη

- Χρήση του συμβολισμού:

  <όνομα-σχέσης>.<όνομα-γνωρίσματος>

- 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

✓ Οι μεταβλητές πλειάδων (AS στο FROM) είναι ιδιαίτερα χρήσιμες

Πράξεις Συνόλου

Πράξεις Συνόλου

Πράξεις:

- UNION (ένωση)
- INTERSECT (τομή)
- EXCEPT (διαφορά)

εφαρμόζονται σε συμβατές σχέσεις.

Γενική Σύνταξη

(SELECT
FROM
WHERE)

UNION/INTERSECT/EXCEPT

(SELECT
FROM
WHERE)

Ένωση

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Τα ονόματα των ηθοποιών που έπαιξαν σε ταινίες του 2006 ή του 2007

```
(SELECT Name
FROM Plays
WHERE Year = 2006)
UNION
(SELECT Name
FROM Plays
WHERE Year = 2007);
```

Ένωση

Απαλοιφή διπλών εμφανίσεων (γίνεται ένα select distinct στο αποτέλεσμα)

εκτός αν χρησιμοποιηθεί το UNION ALL (που απλώς συσσωρεύει τους πίνακες)

Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Ένωση

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
(SELECT Name
FROM Plays
WHERE Year = 2006)
UNION ALL
(SELECT Name
FROM Plays
WHERE Year = 2007);
```

Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Ένωση

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
(SELECT DISTINCT Name
FROM Plays
WHERE Year = 2006)
UNION ALL
(SELECT Name
FROM Plays
WHERE Year = 2007);
```

Τομή

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Τα ονόματα των ηθοποιών που έπαιξαν σε ταινίες του 2006 και του 2007

```
(SELECT Name
FROM Plays
WHERE Year = 2006)
INTERSECT
(SELECT Name
FROM Plays
WHERE Year = 2007);
```

INTERSECT ALL
Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Διαφορά

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
(SELECT Name
FROM Plays
WHERE Year = 2006)
EXCEPT (ή MINUS)
(SELECT Name
FROM Plays
WHERE Year = 2007);
```

EXCEPT ALL

Μέγιστος αριθμός πολλαπλών εμφανίσεων;

Παράδειγμα

```
(SELECT *
 FROM BAG1)
UNION ALL
(SELECT *
 FROM BAG2);
```

BAG1

Fruit
apple
apple
orange
orange

BAG2

Fruit
apple
orange
orange

Παράδειγμα

```
(SELECT *
 FROM BAG1)
INTERSECT ALL
(SELECT *
 FROM BAG2);
```

BAG1

Fruit
apple
apple
orange
orange

BAG2

Fruit
apple
orange
orange

Παράδειγμα

```
(SELECT *
 FROM BAG1)
EXCEPT ALL
(SELECT *
 FROM BAG2);
```

BAG1

Fruit
apple
apple
orange
orange

BAG2

Fruit
apple
orange
orange

Παραδείγματα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

1. Ηθοποιούς που δεν έπαιξαν σε έγχρωμη ταινία
2. Τις ταινίες (τίτλο) με τον ίδιο τίτλο που γυρίστηκαν το 2005 και το 2006 (δώστε δυο ερωτήσεις μια με πράξη συνόλου και μια χωρίς)

Επανάληψη

Πράξεις:

- UNION
- INTERSECT
- EXCEPT (minus)

εφαρμόζονται σε συμβατές σχέσεις (ΠΡΟΣΟΧΗ: πρακτικά τα ΙΔΙΑ ΓΝΩΡΙΣΜΑΤΑ (ίδιο αριθμό και τύπο γνωρισμάτων) στα δύο select)

Σύνταξη,

(SELECT-FROM-WHERE) UNION/INTERSECT/EXCEPT (SELECT-FROM-WHERE)

Απαλοιφή διπλών εμφανίσεων, εκτός αν χρησιμοποιηθεί το UNION/INTERSECT/EXCEPT ALL

Υποερωτήσεις

Υποερωτήσεις

Η SQL επιτρέπει το φώλιασμα υπο-ερωτήσεων.

Μια υπο-ερώτηση είναι μια έκφραση SFW που χρησιμοποιείται *μέσα* σε μια άλλη SFW ερώτηση ως συνθήκη στο WHERE.

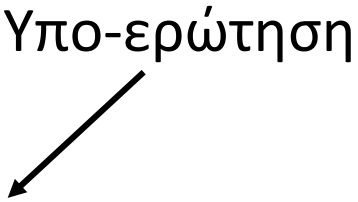
Σύνταξη

SELECT ...

FROM ...

WHERE

<τελεστής> (SELECT ...
FROM ...
WHERE ...);



Η εσωτερική (φωλιασμένη) υπο-ερώτηση υπολογίζεται για κάθε γραμμή (πλειάδα) της εξωτερικής ερώτησης

Στη συνέχεια θα δούμε τι μπορεί να είναι ο τελεστής

Ο τελεστής in (not in)

Ελέγχει αν μια *πλειάδα* ανήκει (δεν ανήκει) σε ένα σύνολο από πλειάδες που έχουν προκύψει από μια έκφραση SFW.

Γενική δομή:

```
SELECT ...
FROM ...
WHERE
  T IN (NOT IN)   (SELECT...
                   FROM ...
                   WHERE   ...  );
```

T: πλειάδα

Ο τελεστής in (not in)

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
SELECT DISTINCT  Actor.Name
FROM  Actor
WHERE Actor.Name  NOT IN
                    (SELECT Name
                     FROM Plays);
```

Ο τελεστής in (not in)

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
SELECT DISTINCT Plays.Name
FROM  Plays
WHERE (Plays.Title, Plays.Year)  IN
```

```
(SELECT Movie.Title, Movie.Year
 FROM   Movie
 WHERE  Type = 'Ασπρόμαυρη');
```

Ο τελεστής in (not in)

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Τον τίτλο όλων των ταινιών με διάρκεια πάνω από 100 λεπτά για τις οποίες υπάρχει ταινία με το ίδιο τίτλο και διάρκεια μικρότερη από 60 λεπτά

```
SELECT DISTINCT Title
FROM  Movie
WHERE Duration > 100
      AND Title IN
```

```
(SELECT Title
 FROM   Movie
 WHERE  Duration < 60);
```

(1) Η ίδια ερώτηση με πράξη συνόλου και με συνένωση

(2) Τροποποίηση της ερώτησης με το IN ώστε η ταινία με διάρκεια < 60 *να είναι διαφορετικού είδους*

Ο τελεστής in (not in)

Μπορεί να χρησιμοποιηθεί και με *enumerated* σύνολα

Παράδειγμα: Τους τίτλους όλων των ταινιών που δεν γυρίστηκαν το 2006 και το 2007.

```
SELECT Title
FROM Movie
WHERE Year NOT IN (2006, 2007);
```

Σύγκριση με (τιμές) συνόλου: any

Ο τελεστής any (some) έχει τη σημασία του *τουλάχιστον ένα* από ένα σύνολο

Γενική δομή:

```
SELECT ...
FROM ...
WHERE
  T > ANY   (SELECT ...
             FROM  ...
             WHERE  ... );
```

T: πλειάδα

any

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Τους τίτλους όλων των ταινιών που γυρίστηκαν αργότερα από τουλάχιστον μια ασπρόμαυρη ταινία

```
SELECT DISTINCT Title
FROM Movie
WHERE Year >ANY      (SELECT  Year
                       FROM Movie
                      WHERE Type = 'Ασπρόμαυρη');
```

any

Επίσης:

<ANY (SOME),

<=ANY (SOME),

>=ANY (SOME),

=ANY (SOME) (ισοδ. του IN)

<>ANY (όχι ισοδ. του NOT IN)

Σύγκριση με (τιμές) συνόλου: all

Ο τελεστής ALL έχει τη σημασία από *όλα* τα στοιχεία ενός συνόλου

Παράδειγμα: Τους τίτλους όλων των ταινιών που γυρίστηκαν αργότερα από όλες τις ασπρόμαυρες ταινίες

```
SELECT DISTINCT Title
FROM Movie
WHERE Year >ALL      (SELECT  Year
                       FROM Movie
                      WHERE Type = 'Ασπρόμαυρη');
```

all

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Τι υπολογίζει το παρακάτω;

```
SELECT Name
FROM Actor
WHERE Year-of-Birth <=ALL (SELECT Year-of-Birth
                           FROM Actor, Plays
                           WHERE Actor.Name = Plays.Name
                           AND Title = 'Μανταλένα');
```

all

Επίσης:

<ALL,

<=ALL,

>=ALL,

=ALL,

<>ALL (ισοδ. του NOT IN)

Ο τελεστής exists (not exists)

Έλεγχος για άδεια σχέση:
Ο τελεστής EXISTS (NOT EXISTS): επιστρέφει true ανν η υποερώτηση δεν είναι κενή (είναι κενή)

Γενική δομή:

```
SELECT ...
FROM ...
WHERE
  EXISTS (NOT EXISTS)   (SELECT ...
                         FROM ...
                         WHERE   ...  );
```

Ο τελεστής exists (not exists)

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Τι επιστρέφει η παρακάτω ερώτηση;

```
SELECT T.Title, T.year
FROM   Movie as T
WHERE  T.type = 'Ασπρόμαυρη' AND
       EXISTS    (SELECT *
                  FROM Plays
                  WHERE Plays.Title = T.Title AND Play.Year = T.year);
```

Ο τελεστής unique (not unique)

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Οι ηθοποιοί που έχουν παίξει τουλάχιστον σε δύο ταινίες

```
SELECT  Name
FROM Plays AS T
WHERE  NOT UNIQUE (SELECT R.Name
                   FROM Plays AS R
                   WHERE T.Name = R.Name);
```

```
SELECT  Name          (θα το δούμε στη συνέχεια)
FROM Plays
GROUP BY Name
HAVING COUNT(*) > 1;
```

Ο τελεστής exists (not exists)

Ο τελεστής NOT EXISTS μπορεί να χρησιμοποιηθεί για έλεγχο αν *η σχέση Α περιέχει τη σχέση Β* (σχέση υπερσυνόλου/υποσυνόλου)

NOT EXISTS (B EXCEPT A)

True if and only if $A \supseteq B$

Παράδειγμα: Οι ηθοποιοί που έχουν παίξει σε <u>όλες</u> τις ταινίες που έχει παίξει ο George Clooney

Ο τελεστής exists (not exists)

NOT EXISTS (B EXCEPT A)

True if and only if $A \supseteq B$

Παράδειγμα: Οι ηθοποιοί που έχουν παίξει σε <u>όλες</u> τις ταινίες που έχει παίξει ο George Clooney

Παράδειγμα Διαίρεσης

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Οι ηθοποιοί που έχουν παίξει σε όλες τις ταινίες που έχει παίξει ο George Clooney

B: όλες οι ταινίες του George Clooney

A: όλες οι ταινίες του συγκεκριμένου ηθοποιού

NOT EXISTS (B EXCEPT A)

```
SELECT DISTINCT S.Name
FROM Plays AS S
WHERE NOT EXISTS   ((SELECT  Title, Year                    B
                     FROM  Plays
                     WHERE Name = 'George Clooney')
                    EXCEPT
                    (SELECT Title, Year                     A
                     FROM Plays as R
                     WHERE  R.Name = S.Name));
```

υπολογισμός για κάθε S

Τέτοιου είδους μεταβλητές δεν υπάρχουν στη σχεσιακή άλγεβρα

Παράδειγμα: Διαίρεση

Τις πίτσες που έχουν όλα τα συστατικά που αρέσουν στον Δημήτρη

Pizza(Name, Ingredient)

Likes(Student, Ingredient)

Θέλουμε τις πίτσες που τα συστατικά τους είναι υπερσύνολο των συστατικών που αρέσουν στο Δημήτρη

A: Συστατικά πίτσας Π

B: Συστατικά που αρέσουν στο Δημήτρη

NOT EXISTS (B EXCEPT A)

Ο τελεστής unique (not unique)

Έλεγχος για Διπλές Εμφανίσεις

Ο τελεστής UNIQUE: επιστρέφει true ανν η υποερώτηση δεν έχει πολλαπλές όμοιες πλειάδες – not unique

Γενική δομή:

```
SELECT ...
FROM  ...
WHERE
  UNIQUE (NOT UNIQUE)   (SELECT ...
                          FROM ...
                          WHERE   ... );
```

Μπορεί να χρησιμοποιηθεί για να ελεγχθεί αν το αποτέλεσμα είναι σύνολο ή πολυσύνολο

Ο τελεστής unique (not unique)

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Οι ηθοποιοί που έχουν παίξει ακριβώς σε μια ταινία

```
SELECT  Name
FROM Plays AS T
WHERE  UNIQUE (SELECT R.Name
                    FROM Plays AS R
                    WHERE T.Name = R.Name);
```

```
SELECT  Name          (θα το δούμε στη συνέχεια)
FROM Plays
GROUP BY Name
HAVING COUNT(*) = 1;
```

Επανάληψη

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παραδείγματα

SELECT Movie.Title FROM Movie
WHERE Duration >=All (SELECT Duration FROM Movie WHERE Type = 'Εγχρωμη');

SELECT Movie.Title FROM Movie
WHERE Duration >SOME (SELECT Duration FROM Movie WHERE Type = 'Εγχρωμη');

SELECT Movie.Title FROM Movie
WHERE Duration IN (SELECT Duration FROM Movie WHERE Type = 'Εγχρωμη');

Επανάληψη

Η SQL επιτρέπει το φώλιασμα υπο-ερωτήσεων.

Μια υπο-ερώτηση είναι μια έκφραση select-from-where που χρησιμοποιείται μέσα σε μια άλλη ερώτηση.

Γενική δομή ως συνθήκη στο WHERE:

```
SELECT ...
FROM  ...
WHERE  <x>
            (SELECT ...
             FROM ...
             WHERE ...  );
```

<x> μπορεί να είναι

T {=, <, <=, >, >=, <>} any(some), all

T in

exists, unique

(όπου *T* πλειάδα)

Δηλαδή διατυπώνονται ως συνθήκες στο where

Υπολογισμός της υπο-ερώτησης για κάθε γραμμή (πλειάδα) της εξωτερικής ερώτησης

R	
A	B
2	3
3	5
7	8

S	
C	D
4	9
7	8
4	2
2	7
1	9

SELECT DISTINCT D FROM S WHERE C NOT IN (SELECT A FROM R);

Το αποτέλεσμα έχει
A. 1 πλειάδα
B. 2 πλειάδες
C. 3 πλειάδες
D. 4 πλειάδες

SELECT * FROM S WHERE B IN (SELECT D FROM R);

Το αποτέλεσμα έχει
A. 1 πλειάδα
B. 2 πλειάδες
C. 3 πλειάδες
D. 4 πλειάδες

R	
A	B
2	3
3	5
7	8

S	
C	D
4	9
7	8
4	2
2	7
1	9

SELECT DISTINCT D FROM S WHERE C >= ANY (SELECT A FROM R);

Το αποτέλεσμα έχει
A. 1 πλειάδα
B. 2 πλειάδες
C. 3 πλειάδες
D. 4 πλειάδες

SELECT * FROM S WHERE B <> ANY (SELECT D FROM R);

Το αποτέλεσμα έχει
A. 1 πλειάδα
B. 2 πλειάδες
C. 3 πλειάδες
D. 4 πλειάδες

R	
A	B
2	3
3	5
7	8

S	
C	D
4	9
7	8
4	2
2	7
1	9

```
SELECT DISTINCT D  FROM S WHERE C >= ALL (SELECT A FROM R);
```

Το αποτέλεσμα έχει
A. 1 πλειάδα
B. 2 πλειάδες
C. 3 πλειάδες
D. 4 πλειάδες

```
SELECT *  FROM S WHERE A <> ALL (SELECT D FROM R);
```

Το αποτέλεσμα έχει
A. 1 πλειάδα
B. 2 πλειάδες
C. 3 πλειάδες
D. 4 πλειάδες

Επανάληψη

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παραδείγματα

```
SELECT A.Name
FROM Actor AS A
WHERE EXISTS (SELECT *
              FROM Actor AS S
              WHERE A.Year-of-Birth = S.Year-of-Birth AND
              A.Name <> S.Name);
```

Το ίδιο με NOT UNIQUE?

Συναθροιστικές Συναρτήσεις

Συναθροιστικές Συναρτήσεις

Η SQL έχει 5 built-in συναθροιστικές συναρτήσεις (aggregate functions):

Μέσος όρος: AVG(A) (μόνο σε αριθμούς) Α γνώρισμα
Ελάχιστο: MIN(A)
Μέγιστο: MAX(A)
Άθροισμα: SUM(A) (μόνο σε αριθμούς)
Πλήθος: COUNT(A)

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Μέση διάρκεια όλων των έγχρωμων ταινιών

```
SELECT AVG(Duration)
FROM Movie
WHERE Type = 'Εγχρωμη';
```

Το αποτέλεσμα είναι μια σχέση με ένα γνώρισμα και μια γραμμή *[μπορούμε να δώσουμε όνομα στο γνώρισμα χρησιμοποιώντας το AS]*

- Εμφανίζονται στο SELECT

Συναθροιστικές Συναρτήσεις

Παράδειγμα: Μέγιστη διάρκεια όλων των έγχρωμων ταινιών και την ταινία με τη μεγαλύτερη διάρκεια!!

```
SELECT Title, Year, MAX(Duration)
FROM Movie
WHERE Type = 'Εγχρωμη';
```

Αν το SELECT συναθροιστική, τότε μόνο συναθροιστικές, - *εκτός αν υπάρχει group by - δηλαδή δεν μπορούμε να προβάλουμε και άλλα γνωρίσματα σχέσεων*

Παράδειγμα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

SERVES(Place, Name)

```
SELECT COUNT(Name)
FROM PIZZA;
```

```
SELECT COUNT(DISTINCT Name)
FROM PIZZA;
```

```
SELECT Name, COUNT(*)
FROM PIZZA
GROUP BY Name;
```

Παράδειγμα

PIZZA

Name	Ingredient
Vegetarian	μανιτάρι
Vegetarian	ελιά
Χαβάη	ανανάς
Χαβάη	ζαμπόν
Σπέσιαλ	ζαμπόν
Σπέσιαλ	μπέικον
Σπέσιαλ	μανιτάρι
Ελληνική	ελιά

SERVES

Place	Name
Roma	Vegetarian
Roma	Σπέσιαλ
Napoli	Vegetarian
Napoli	Ελληνική
Pizza-Express	Χαβάη
Pizza-Express	Σπέσιαλ
Pizza-Express	Ελληνική
Pizza-Place	Σπέσιαλ

LIKES

Student	Ingredient
Δημήτρης	μανιτάρι
Κώστας	ζαμπόν
Μαρία	ελιά
Κατερίνα	μανιτάρι
Μαρία	ζαμπόν
Δημήτρης	μπέικον
Μαρία	ανανάς

Παράδειγμα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

SERVES(Place, Name)

- Πόσα συστατικά που αρέσουν στο Δημήτρη έχει κάθε πίτσα;
- Πόσες πίτσες με συστατικά που αρέσουν στον Δημήτρη σερβίρει κάθε μαγαζί;

Συναθροιστικές Συναρτήσεις

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

```
SELECT SUM(DISTINCT  Duration)
FROM   Movie;
```

Συναθροιστικές Συναρτήσεις

Για να μετρήσουμε πόσες πλειάδες έχει μια σχέση:

```
SELECT COUNT(*)
FROM Movie;
```

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

R

A	B	C
1	5	6
2	3	2
1	9	3
7	2	9
7	8	3
1	5	2
4	2	1
2	3	3
4	1	8

```
SELECT COUNT(C)
FROM R;
```

```
SELECT COUNT(DISTINCT C)
FROM R;
```

```
SELECT COUNT(*)
FROM R;
```

```
SELECT SUM(C)
FROM R;
```

```
SELECT SUM(DISTINCT C)
FROM R;
```

```
SELECT AVG(C)
FROM R;
```

```
SELECT AVG(DISTINT C)
FROM R;
```

```
SELECT MIN(C)
FROM R;
```

```
SELECT MAX(C)
FROM R;
```

Συναθροιστικές Συναρτήσεις: group by

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

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

Παράδειγμα 1: Μέση διάρκεια ταινίας ανά είδος

```
SELECT Type, avg(Duration)
FROM Movie
GROUP BY Type;
```

Στο SELECT και η τιμή του γνωρίσματος του GROUP BY

Παράδειγμα

R

A	B	C
1	5	6
2	3	2
1	9	3
7	2	9
7	8	3
1	5	2
4	2	1
2	3	3
4	1	8

```
SELECT A, MAX(C)
FROM R
GROUP BY A;
```

```
SELECT A, MAX(C)
FROM R
WHERE A < B
GROUP BY A;
```

Συναθροιστικές Συναρτήσεις: group by

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

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

Παράδειγμα: Τον αριθμό ταινιών που έπαιξε κάθε ηθοποιός

Συναθροιστικές Συναρτήσεις: group by

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Η ομαδοποίηση μπορεί να γίνει ως προς περισσότερα του ενός πεδία.

Παράδειγμα

R

A	B	C
1	5	6
2	3	2
1	9	3
7	2	9
7	8	3
1	5	2
4	2	1
2	3	3
4	1	8

```
SELECT A, B, MAX(C)
FROM R
GROUP BY A, B;
```

Συναθροιστικές Συναρτήσεις: group by

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Τι υπολογίζει η παρακάτω ερώτηση;

```
SELECT Title, Year,  COUNT(Name)
FROM Plays
GROUP BY Title, Year;
```

Συναθροιστικές Συναρτήσεις: having

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

```
SELECT  Year,  COUNT(Title)
FROM Movie
GROUP BY Year
HAVING AVG(Duration) > 100;
```

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

Συναθροιστικές Συναρτήσεις

Όταν 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 και επιλέγονται όσες ικανοποιούν τη συνθήκη

Συναθροιστικές Συναρτήσεις

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

Παράδειγμα: Αριθμό ταινιών που έπαιξε κάθε ηθοποιός που γεννήθηκε μετά το 1987 αν αυτός ο αριθμός είναι μεγαλύτερος του 5

```
SELECT   Actor.Name, count(*)
FROM Plays, Actor
WHERE  Plays.Name = Actor.Name   AND  Year-of-Birth> 1987
GROUP BY Actor.Name
HAVING COUNT(*) >= 5;
```

4
1
2
3

Παράδειγμα

R

A	B	C
1	5	6
2	3	2
1	9	3
7	2	9
7	8	3
1	5	2
4	2	1
2	3	3
4	1	8

```
SELECT A, MAX(C)
FROM R
WHERE A < B
GROUP BY A
HAVING MAX(B) > 2;
```

Παράδειγμα

R

A	B	C
1	5	6
2	3	2
1	9	3
7	2	9
7	8	3
1	5	2
4	2	1
2	3	3
4	1	8

Μια πλειάδα στην οποία εμφανίζεται η μεγαλύτερη τιμή του B (δύο τρόποι – ORDER BY και MAX)

Επανάληψη

Μέσος όρος: AVG (μόνο σε αριθμούς)
Ελάχιστο: MIN
Μέγιστο: MAX
Άθροισμα: SUM (μόνο σε αριθμούς)
Πλήθος: COUNT

- Αν θέλουμε να απαλείψουμε διπλές εμφανίσεις χρησιμοποιούμε τη λέξη-κλειδί DISTINCT στην αντίστοιχη έκφραση.
- 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
- Μπορούμε να εφαρμόσουμε μια συνθήκη σε μια συγκεκριμένη ομάδα από πλειάδες χρησιμοποιώντας το HAVING. Η συνθήκη του HAVING εφαρμόζεται αφού σχηματιστούν οι ομάδες και υπολογιστούν οι συναθροιστικές συναρτήσεις
- Οι null τιμές αγνοούνται πλην του count(*)

Βασική Δομή Ερώτησης

SELECT $A_{i1}, A_{i2}, .., A_{in,} \ldots$, avg, …
FROM $R_1, R_2, \ldots R_m$
WHERE P
GROUP BY $A_{i1}, A_{i2}, \ldots, A_{in}$
HAVING P
ORDER BY $A_{j1}, A_{j2}, \ldots, A_{jk}$

Συνενώσεις

Συνένωση (join)

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

Γενική σύνταξη:

<όνομα-σχέσης1> <τύπος-συνένωσης> <όνομα-σχέσης2>
on <συνθήκη-συνένωσης>

Τύποι Συνένωσης:

- NATURAL JOIN
- [INNER] JOIN
- LEFT [OUTER] JOIN: αριστερή εξωτερική συνένωση
- RIGHT [OUTER] JOIN: δεξιά εξωτερική συνένωση
- FULL [OUTER] JOIN: πλήρης εξωτερική συνένωση

R

A	B
1	red
3	blue
4	green

```
SELECT *
FROM R INNER JOIN S ON R.A = S.A;
```

S

A	D
1	car
3	bicycle
1	bicycle
8	car

```
SELECT *
FROM R NATURAL JOIN S;
```

```
SELECT *
FROM R LEFT OUTER JOIN S ON R.A = S.A;
```

R

A	B
1	red
3	blue
4	green

```
SELECT *
FROM R NATURAL LEFT OUTER JOIN S;
```

S

A	D
1	car
3	bicycle
1	bicycle
5	bicycle
8	car

```
SELECT *
FROM R RIGHT OUTER JOIN S ON R.A = S.A;
```

```
SELECT *
FROM R FULL OUTER JOIN S ON R.A = S.A;
```

Παράδειγμα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

```
SELECT DISTINCT LIKES. Student, PIZZA.Name
FROM (LIKES LEFT OUTER JOIN PIZZA
    ON LIKES.Ingredient  = PIZZA.Ingredient);
```

Παράδειγμα

PIZZA

Name	Ingredient
Vegetarian	μανιτάρι
Vegetarian	ελιά
Χαβάη	ανανάς
Χαβάη	ζαμπόν
Σπέσιαλ	ζαμπόν
Σπέσιαλ	μπέικον
Σπέσιαλ	μανιτάρι
Ελληνική	ελιά
Γιαννιώτικη	μετσοβόνε

LIKES

Student	Ingredient
Δημήτρης	μανιτάρι
Κώστας	ζαμπόν
Μαρία	ελιά
Κατερίνα	μανιτάρι
Μαρία	ζαμπόν
Δημήτρης	μπέικον
Μαρία	ανανάς
Ανδρόνικος	αντσούγια

Φυσική Συνένωση (natural join)

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

```
SELECT DISTINCT LIKES.Student, PIZZA.Name
FROM PIZZA NATURAL JOIN LIKES;
```

```
SELECT DISTINCT LIKES.Student, PIZZA.Name
FROM  PIZZA, LIKES
WHERE PIZZA.Ingredient = LIKES.Ingredient;
```

Παράδειγμα

LIKES

Student	Ingredient
Δημήτρης	μανιτάρι
Κώστας	ζαμπόν
Μαρία	ελιά
Κατερίνα	μανιτάρι
Μαρία	ζαμπόν
Δημήτρης	μπέικον
Μαρία	ανανάς
Ανδρόνικος	αντσούγια

PIZZA

Name	Ingredient
Vegetarian	μανιτάρι
Vegetarian	ελιά
Χαβάη	ανανάς
Χαβάη	ζαμπόν
Σπέσιαλ	ζαμπόν
Σπέσιαλ	μπέικον
Σπέσιαλ	μανιτάρι
Ελληνική	ελιά
Γιαννιώτικη	μετσοβόνε

Παράδειγμα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

```
SELECT DISTINCT LIKES. Student, PIZZA.Name
FROM (LIKES RIGHT OUTER JOIN PIZZA
    ON LIKES.Ingredient  = PIZZA.Ingredient);
```

Παράδειγμα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

```
SELECT DISTINCT LIKES. Student, PIZZA.Name
FROM  (LIKES INNER JOIN PIZZA
    ON PIZZA.Ingredient  = LIKES.Ingredient);
```

```
SELECT DISTINCT LIKES. Student, PIZZA.Name
FROM LIKES, PIZZA
WHERE PIZZA.Ingredient  = LIKES.Ingredient;
```

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
SELECT DISTINCT Movie.Name
FROM  Movie, Plays
WHERE Plays.Tite = Movie.Title AND Plays.Year = Movie.Year and
Type = 'Ασπρόμαυρη';

SELECT DISTINCT Movie.Name
FROM  (Movie JOIN Plays ON Plays.Tite = Movie.Title AND Plays.Year =
Movie.Year)
WHERE Type = 'Ασπρόμαυρη';

SELECT DISTINCT Movie.Name
FROM  Movie NATURAL JOIN Plays
WHERE Type = 'Ασπρόμαυρη';
```

SFW στο FROM

Μπορούμε να έχουμε μια SFW ερώτηση στο FROM

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

```
SELECT DISTINCT P.Name
FROM PIZZA AS P,
        ((SELECT Ingredient
        FROM LIKES
        WHERE Student = 'Δημήτρης')
                EXCEPT
        (SELECT Ingredient
        FROM LIKES
        WHERE Student = 'Μαρία'))    AS T
WHERE P.Ingredient = T.Ingredient;
```

With

WITH <όνομα--query> [<λίστα ονομάτων-γνωρισμάτων>] (**AS** <SELECT-FROM-WHERE ερώτηση>)

SELECT …

- Ορίζεται όπως μια view αλλά δεν είναι ανεξάρτητη πρέπει να ακολουθεί SFW ερώτηση και μπορεί να χρησιμοποιηθεί μόνο σε αυτήν (το scope είναι η ερώτηση που ακολουθεί)
- Common Table Expressions (CTEs) – ονοματιζόμενα προσωρινά αποτελέσματα

WITH cte_name (column_list) **AS** (**query**)
SELECT * **FROM** cte_name;

- Μπορεί να έχουμε πολλαπλούς ορισμούς στο ίδιο WITH χωρισμένους με κόμμα

SFW – WITH, FROM

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

```
SELECT DISTINCT P.Name
FROM PIZZA AS P,
            ((SELECT Ingredient
            FROM LIKES
            WHERE Student = 'Δημήτρης')
                    EXCEPT
            (SELECT Ingredient
            FROM LIKES
            WHERE Student = 'Μαρία'))    AS T
WHERE P.Ingredient = T.Ingredient;
```

```
WITH T(Ingredient) AS ((SELECT Ingredient
                        FROM LIKES
                        WHERE Student = 'Δημήτρης')
                                EXCEPT
                        (SELECT Ingredient
                        FROM LIKES
                        WHERE Student = 'Μαρία'))
SELECT DISTINCT P.Name
FROM PIZZA AS P, T
WHERE P.Ingredient = T.Ingredient;
```

With

Movie(Title, Year, Duration, Type)

Plays(Name, Title, Year)

Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
WITH Actor1(aName) AS (SELECT Name FROM Actor),
Actor2(aName) AS (SELECT Spouse-Name FROM Actor)

SELECT * FROM Actor1 INTERSECT Actor2;
```

Παράδειγμα

Με FROM, WITH

R

A	B	C
1	5	6
2	3	2
1	9	3
7	2	9
7	8	3
1	5	2
4	2	1
2	3	3
4	1	8

Τη μέση τιμή του B στις πλειάδες που έχουν την μικρότερη τιμή του A

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

SERVES(Place, Name)

1. Πόσα μαγαζιά υπάρχουν στη βάση δεδομένων

2. Το όνομα του μαγαζιού που σερβίρει τις περισσότερες πίτσες

3. Τις πίτσες που έχουν τουλάχιστον 3 συστατικά που αρέσουν σε φοιτητές

   Τα μαγαζιά που σερβίρουν αυτές τις πίτσες

4. Τις πίτσες που δεν έχουν **κανένα** συστατικό που να αρέσει σε φοιτητές
   το 4 με πράξη συνόλων

5. Για κάθε πίτσα τον αριθμό των συστατικών που περιέχει και αρέσουν σε φοιτητές (για τις πίτσες που δεν έχουν κανένα συστατικά να εμφανίζεται ο αριθμός 0)

6. Τους φοιτητές (ζεύγη) που τους αρέσουν ακριβώς τα ίδια συστατικά (να μην εμφανίζονται συμμετρικά ζεύγη και ζεύγη με τον φοιτητή και τον εαυτό του)

Γλώσσα Ενημερώσεις Δεδομένων

Εισαγωγή

- Γλώσσα Ορισμού (του σχήματος)
- Γλώσσα Χειρισμού Δεδομένων (ΓΧΔ)
  - Γλώσσα Τροποποίησης Δεδομένων (εισαγωγή, διαγραφή, ενημέρωση πλειάδων)
  - Γλώσσα Ερωτήσεων (Query Languages)

Τροποποίηση ΒΔ

Τροποποιήσεις

1. Διαγραφή
2. Εισαγωγή
3. Ενημέρωση

Οι εντολές αυτές μεταβάλλουν το στιγμιότυπο της βάσης δεδομένων (δηλαδή, το περιεχόμενο των πινάκων)

Δείτε και τις σχετικές διαφάνειες προηγούμενου μαθήματος

Εισαγωγή δεδομένων

Για να εισάγουμε δεδομένα σε μια σχέση είτε

(α) προσδιορίζουμε την πλειάδα,

INSERT INTO R(A_1, …, A_n) VALUES (v_1, …, v_n);

είτε

(β) γράφουμε μια ερώτηση που το αποτέλεσμα της εισάγεται στη σχέση.

INSERT INTO R(A_1, …, A_n) SELECT-FROM-WHERE

Εισαγωγή δεδομένων

PIZZA(Name, Ingredient)

LIKEΣ(Student, Ingredient)

Παράδειγμα

Εισαγωγή μιας πίτσας στη ΠΙΤΣΑ με όνομα «Κατερίνας-special» με συστατικά τα συστατικά που αρέσουν στη φοιτήτρια Κατερίνα

```
INSERT INTO PIZZA(PIZZA.Name, PIZZA.Ingredient)
        SELECT  `Κατερίνας-Special', LIKES.Ingredient
        FROM LIKES
        WHERE LIKES.Student = 'Κατερίνα';
```

Διαγραφή δεδομένων

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.

Διαγραφή δεδομένων

- Στο FROM μόνο μια σχέση, αλλά στη συνθήκη του WHERE μπορεί να εμφανίζονται και άλλες

- Σβήνονται «ολόκληρες» πλειάδες

- Αν υπάρχουν παραπάνω από μια πλειάδες που ικανοποιούν τη συνθήκη, δεν υπάρχει τρόπος να διακρίνουμε τις πλειάδες, δηλαδή να σβήσουμε κάποιες

- Πρώτα, υπολογίζεται η συνθήκη του WHERE και μετά διαγράφονται οι πλειάδες που ικανοποιούν τη συνθήκη

```
DELETE FROM Plays
WHERE Title  IN  (SELECT Title
                  FROM Movie
                  WHERE Type = 'Εγχρωμη');
```

Ενημέρωση

```
UPDATE R
SET Attr = New_Value
WHERE  P
```

Παράδειγμα: Αύξηση τις διάρκειας κάθε ταινίας κατά 10 λεπτά για όλες τις ταινίες με διάρκεια < 100

```
UPDATE Movie
SET Duration = Duration + 10
WHERE Duration < 100;
```

Ενημέρωση

Όπως και για τη διαγραφή:

- Στο UPDATE μόνο μια σχέση, αλλά στη συνθήκη του WHERE μπορεί να εμφανίζονται και άλλες

- Αν υπάρχουν παραπάνω από μια πλειάδες που ικανοποιούν τη συνθήκη, δεν υπάρχει τρόπος να διακρίνουμε τις πλειάδες, δηλαδή να ενημερώσουμε κάποιες

- Πρώτα, υπολογίζεται η συνθήκη του WHERE και μετά ενημερώνονται οι πλειάδες που ικανοποιούν τη συνθήκη – δηλαδή, η συνθήκη υπολογίζεται στο τρέχων στιγμιότυπο – όχι στο τροποποιημένο

Επανάληψη

1. Εισαγωγές

INSERT INTO R(A_1, …, A_n) VALUES (v_1, …, v_n)

INSERT INTO R(A1, …, An) SFW

2. Διαγραφές

DELETE FROM R WHERE P

3. Ενημερώσεις/Τροποποιήσεις

UPDATE R
SET Attr = New_Value
WHERE P

Όψεις

Ορισμός Όψεων (εικονικών πινάκων)

Μπορούμε να ορίσουμε μια όψη χρησιμοποιώντας την εντολή:

CREATE VIEW <όνομα--όψης> **AS** <SELECT-FROM-WHERE ερώτηση>

Επίσης, μπορούν να προσδιοριστούν τα ονόματα των γνωρισμάτων άμεσα

CREATE VIEW <όνομα--όψης> (<λίστα ονομάτων-γνωρισμάτων>)
AS <SELECT-FROM-WHERE ερώτηση>

Διαφορά από create table

- **Αποθηκεύετε** ο ορισμός
- Μπορεί να χρησιμοποιηθεί όπου ένας πίνακας, αλλά η όψη (δηλαδή, το περιεχόμενο του πίνακα) *υπολογίζεται εκ νέου* κάθε φορά
- Χρήση: Σε ερωτήματα που υπολογίζονται συχνά ή (κυρίως) για έλεγχο πρόσβασης

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
CREATE VIEW BlackAndWhite AS
  SELECT  Title, Year
  FROM Movie
  WHERE  Type = 'Ασπρόμαυρη';
```

Base relations/tables

Βασική Σχέση

Ενημερώσιμες Όψεις

- Για ενημερώσεις ισχύουν περιορισμοί -- Τροποποιήσεις μέσω όψεων
  - Ενημερώσιμες όψεις - updatable
    - ένα μόνο πίνακα, πρωτεύον κλειδί της βασικής σχέσης και τιμές για όλα τα *not null* γνωρίσματα χωρίς default τιμή (select, project)
  - Υλοποιημένη (materialized) όψη

Παράδειγμα

Movie(Title, Year, Duration, Type)
Plays(Name, Title, Year)
Actor(Name, Address, Year-of-Birth, Spouse-Name)

```
CREATE VIEW ActorStatistics (ActorName, NumbofMovies) AS
  SELECT  Plays,Name, COUNT(*)
  FROM Plays
  GROUP BY Plays.Name;
```

Μη ενημερώσιμη!

Διαγραφή όψης

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

  DROP VIEW <όνομα-όψης>

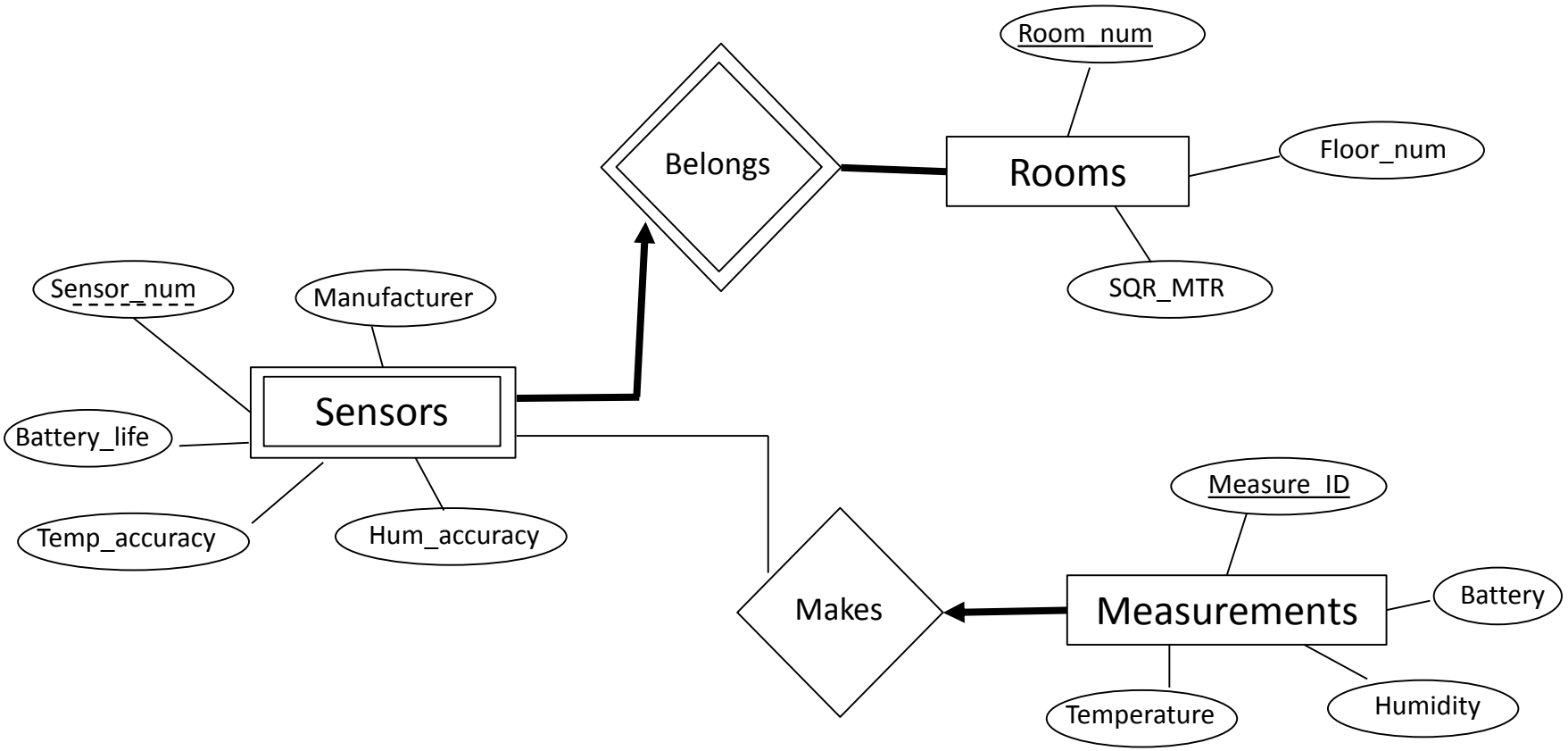
Απαντήσεις 1ου Quiz

Ερώτηση για τη βάση δεδομένων με τους αισθητήρες

Άσκηση

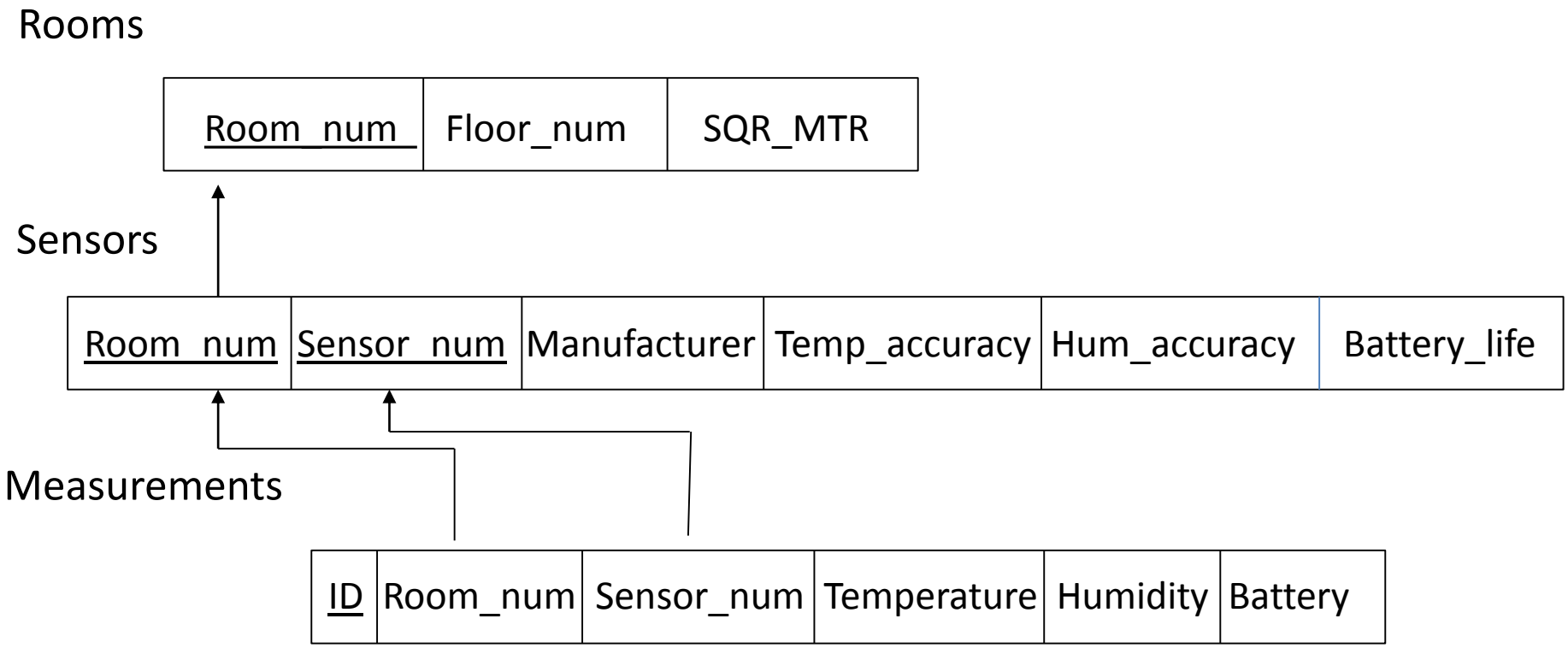
Θέλουμε να κατασκευάσουμε μια βάση δεδομένων στην οποία θα αποθηκεύουμε *αποτελέσματα μετρήσεων από αισθητήρες* που έχουμε εγκαταστήσει στα δωμάτια ενός κτιρίου. Οι αισθητήρες μετρούν θερμοκρασία και ποσοστό υγρασίας.

- Για κάθε *δωμάτιο* έχουμε έναν μοναδικό αριθμό-δωματίου, τον όροφο στον οποίο βρίσκεται και τα τετραγωνικά του μέτρα.
- Ένας *αισθητήρας* χαρακτηρίζεται από τον αριθμό-δωματίου στον οποίο έχει εγκατασταθεί και από έναν αριθμό-αισθητήρα που είναι μοναδικός ανά δωμάτιο (δηλαδή, δεν υπάρχουν αισθητήρες με τον ίδιο αριθμό-αισθητήρα στο ίδιο δωμάτιο). Για κάθε αισθητήρα καταγράφουμε τον κατασκευαστή του, τη μέγιστη διάρκεια ζωής της μπαταρίας του (σε ώρες), την ακρίβεια του (σε ποσοστό) ως αναφορά τη θερμοκρασία, και την ακρίβεια του (σε ποσοστό) ως αναφορά τη υγρασία.
- Για κάθε *μέτρηση*, έχουμε ένα μοναδικό id, τον αισθητήρα που την κατέγραψε και τις δύο τιμές (θερμοκρασία, υγρασία) της μέτρησης και την υπολειπόμενη μπαταρία (ποσοστό).
- Σε κάθε δωμάτιο έχουμε εγκαταστήσει τουλάχιστον έναν αισθητήρα



ολική συμμετοχή
μερική συμμετοχή

Belongs N-1 (από το Sensors στο Rooms)
Makes 1-N (από το Sensors στο Measurements)


Rooms
Room_num
Floor_num
SQR_MTR
Sensors
Room_num
Sensor_num
Manufacturer
Temp_accuracy
Hum_accuracy
Battery_life
Measurements
ID
Room_num
Sensor_num
Temperature
Humidity
Battery

Θεωρείστε ότι στη βάση δεδομένων με τις μετρήσεις από αισθητήρες που είδαμε στο εργαστήριο έχουμε τον περιορισμό ότι ***σε κάθε δωμάτιο μπορούμε να έχουμε μόνο έναν αισθητήρα***. Τι πρέπει να τροποποιηθεί στον ορισμό του πίνακα Sensors;

Επιλέξτε ένα ή περισσότερα:
Ορίζουμε ως UNIQUE το γνώρισμα Sensor_num.
Θέτουμε PRIMARY KEY(Room_num)
Δε χρειάζεται καμία αλλαγή.
Θέτουμε PRIMARY KEY(Sensor_num)

Θεωρείστε ότι στη βάση δεδομένων με τις μετρήσεις από αισθητήρες που είδαμε στο εργαστήριο, **δεν μπορεί να υπάρχουν αισθητήρες με το ίδιο Sensor_num στο ίδιο ή σε διαφορετικά δωμάτια.** Τι πρέπει να αλλάξει στον ορισμό του σχήματος;

Επιλέξτε ένα ή περισσότερα:
Στον πίνακα Sensors δε χρειάζεται το γνώρισμα Room_num.
Στον πίνακα Measurements δε χρειάζεται το γνώρισμα Room_num.
Στον πίνακα Sensors θέτουμε PRIMARY KEY(Sensor_num).
Στον πίνακα Sensors θέτουμε PRIMARY KEY(Room_num).

Θεωρείστε ότι στη βάση δεδομένων με τις μετρήσεις από αισθητήρες που είδαμε στο εργαστήριο, **οι μετρήσεις δεν έχουν μοναδικό ID, αλλά μετρήσεις από διαφορετικούς αισθητήρες μπορεί να έχουν το ίδιο ID.** Τι πρέπει να αλλάξει στον ορισμό του σχήματος;

Επιλέξτε ένα ή περισσότερα:
Στον πίνακα Measurements ορίζουμε PRIMARY KEY(ID, Room_num, Sensor_num)
Δε χρειάζεται καμία αλλαγή.
Στον πίνακα Measurements ορίζουμε PRIMARY KEY(ID, Sensor_num)
Στον πίνακα Measurements ορίζουμε το γνώρισμα Sensor_num ως UNIQUE.

Έστω η βάση δεδομένων με τις μετρήσεις από αισθητήρες που είδαμε στο εργαστήριο. Θεωρείστε την παρακάτω αλλαγή: **ο αριθμός δωματίου δεν είναι μοναδικός, αλλά μπορεί να υπάρχουν δωμάτια με τον ίδιο αριθμό σε διαφορετικούς ορόφους**.

Ποιες από τις παρακάτω τροποποιήσεις του σχήματος είναι σωστές;

Επιλέξτε ένα ή περισσότερα:
Στον πίνακα Sensors πρέπει να προστεθεί το γνώρισμα Floor_num και να έχουμε PRIMARY KEY(Floor_num).
Στον πίνακα Rooms θέτουμε PRIMARY KEY(Floor_num, Room_num). Σ
Στον πίνακα Measurements πρέπει να προστεθεί το γνώρισμα Floor_num και να έχουμε PRIMARY KEY(Floor_num, Room_num, Sensor_Num, ID).
Στον πίνακα Sensors πρέπει να προστεθεί το γνώρισμα Floor_num και να έχουμε PRIMARY KEY(Floor_num, Room_num, Sensor_Num). Σ

Ερώτηση για πληθικότητες της Teaches

Η παρακάτω βάση δεδομένων υλοποιεί δύο τύπους οντοτήτων (O1: Professor, O2: Course) και τη μεταξύ τους σχέση (Teaches).

CREATE TABLE Professor (ProfessorID INT PRIMARY KEY, ProfessorName VARCHAR NOT NULL);

CREATE TABLE Course (CourseID INT PRIMARY KEY, Title VARCHAR NOT NULL, ECTS INT NOT NULL);

CREATE TABLE Teaches (CourseID INT REFERENCES Course (CourseID) ON DELETE CASCADE, ProfessorID INT REFERENCES Professor (ProfessorID) ON DELETE CASCADE,

PRIMARY KEY (CourseID, ProfessorID));

Ένας καθηγητής μπορεί να διδάσκει παραπάνω από ένα μάθημα.
Ένας καθηγητής διδάσκει το πολύ ένα μάθημα.
Ένα μάθημα ανατίθεται το πολύ σε ένα καθηγητή.
Ένα μάθημα μπορεί να ανατεθεί σε πολλούς καθηγητές.

PRIMARY KEY (CourseID));

Ένας καθηγητής διδάσκει το πολύ ένα μάθημα.
Ένας καθηγητής μπορεί να διδάσκει παραπάνω από ένα μάθημα.
Ένα μάθημα μπορεί να ανατεθεί σε πολλούς καθηγητές.
Ένα μάθημα ανατίθεται το πολύ σε ένα καθηγητή.

PRIMARY KEY (ProfessorID));

Ένας καθηγητής διδάσκει το πολύ ένα μάθημα.
Ένας καθηγητής μπορεί να διδάσκει παραπάνω από ένα μάθημα.
Ένα μάθημα ανατίθεται το πολύ σε ένα καθηγητή.
Ένα μάθημα μπορεί να ανατεθεί σε πολλούς καθηγητές.

Η συσχέτιση Teaches είναι 1-N (1-προς-πολλά) από το Professor στο Course. Ποιος ή ποιοι από τους παρακάτω ορισμούς πρωτεύοντος κλειδιού για τον πίνακα Teaches μοντελοποιούν αυτήν την πληθικότητα;

PRIMARY KEY(CourseID)
PRIMARY KEY(ProfessorID)
Αυτός ο περιορισμός πληθικότητας δε μπορεί να εκφραστεί στο σχεσιακό μοντέλο.
PRIMARY KEY(ProfessorID, CourseID

Ερώτηση ξένο κλειδί

Δημιουργούμε μια βάση δεδομένων με τους πίνακες DEPARTMENT και EMPLOYEE ως εξής:

```
CREATE TABLE DEPARTMENT (
DepartmentID INT PRIMARY KEY,
Name VARCHAR NOT NULL
);
CREATE TABLE EMPLOYEE (
   EmployID INT PRIMARY KEY,
   DepartmentID  INT REFERENCES DEPARTMENT(DepartmentID)
);
```

Ο πίνακας DEPARTMENT έχει τις εξής πλειάδες:

DepartmentID	Name
7	Marketing
2	Accounting
1	Production

Ποια από τα ακόλουθα στιγμιότυπα ***δεν μπορεί*** να είναι έγκυρα στιγμιότυπα του πίνακα EMPLOYEE;

EmployeeID	DepartmentID
1010	1
1020	7
1030	4

EmployeeID	DepartmentID
1010	1
1020	NULL
1030	1

EmployeeID	DepartmentID
1010	1
1020	7
1030	1

Ερώτηση insert

Δημιουργούμε μια βάση δεδομένων με τους πίνακες Professor και Student ως εξής:

CREATE TABLE Professor (ProfessorID INT PRIMARY KEY, Name VARCHAR NOT NULL UNIQUE);

CREATE TABLE Student (StudentID INT PRIMARY KEY, Advisor INT DEFAULT NULL REFERENCES Professor(ProfessorID) ON DELETE SET DEFAULT);

Οι πίνακες Professor και Student έχουν τις εξής πλειάδες:

ProfessorID	Name
1	Maria Smith
2	May Jordan
3	Alica Keys

StudentID	Advisor
101	1
220	2
350	3

Ποιες από τις παρακάτω εισαγωγές **δεν** γίνονται δεκτές;

INSERT INTO Student(StudentID, Advisor) VALUES (350, 1);
INSERT INTO Professor(ProfessorID, Name) VALUES (4, 'Maria Smith');
INSERT INTO Student(StudentID, Advisor) VALUES (410, 4);
INSERT INTO Student(StudentID, Advisor) VALUES(410, 3);
INSERT INTO Student(StudentID, Advisor) VALUES (410, NULL);

Ερώτηση πολλαπλά ξένα κλειδιά

Έστω μια βάση δεδομένων με τρεις πίνακες R1, R2 και R3 ορισμένους ως:
CREATE TABLE R1 (A1 INT PRIMARY KEY, B1 INT);
CREATE TABLE R2 (A2 INT PRIMARY KEY, B2 INT, FOREIGN KEY A2 REFERENCES R1(A1) ON DELETE CASCADE ON UPDATE CASCADE);
CREATE TABLE R3 (A3 INT PRIMARY KEY, B3 INT, FOREIGN KEY B3 REFERENCES R2(A2) ON DELETE SET NULL ON UPDATE SET NULL);
Στο τρέχων στιγμιότυπο της βάσης δεδομένων, οι πίνακες είναι όπως παρακάτω:

R1	
A1	B1
1	2
3	3
4	6

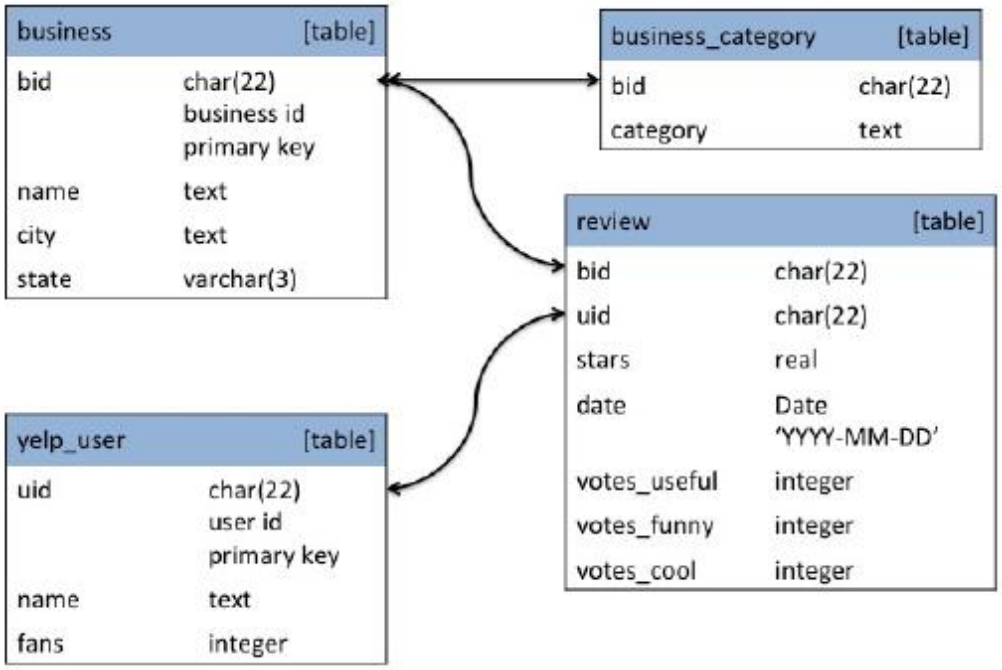
R2	
A2	B2
1	4
4	5

R3	
A3	B3
1	1
2	4
4	4

1. Μετά την εκτέλεση της εντολής UPDATE R1 SET A1 = 2 WHERE B1 = 3, ο πίνακας R2 θα έχει 2 πλειάδες.
2. Η εντολή DELETE FROM R2 έχει ως αποτέλεσμα να σβηστεί το περιεχόμενο των πινάκων R2 και R3. Λ
3. Η εντολή INSERT INTO R2(A2, B2) VALUES (4, 3) δε γίνεται δεκτή.
4. Η εκτέλεση της εντολής DELETE FROM R1 WHERE B1 = 6 δεν επηρεάζει το περιεχόμενο του πίνακα R3. Λ
5. Μετά την εκτέλεση της εντολής DELETE FROM R2 WHERE A1 = 4, ο πίνακας R3 θα έχει 1 πλειάδα. Λ
6. Η εκτέλεση της εντολής DELETE FROM R1 WHERE B1 = 3 δεν επηρεάζει το περιεχόμενο του πίνακα R2.

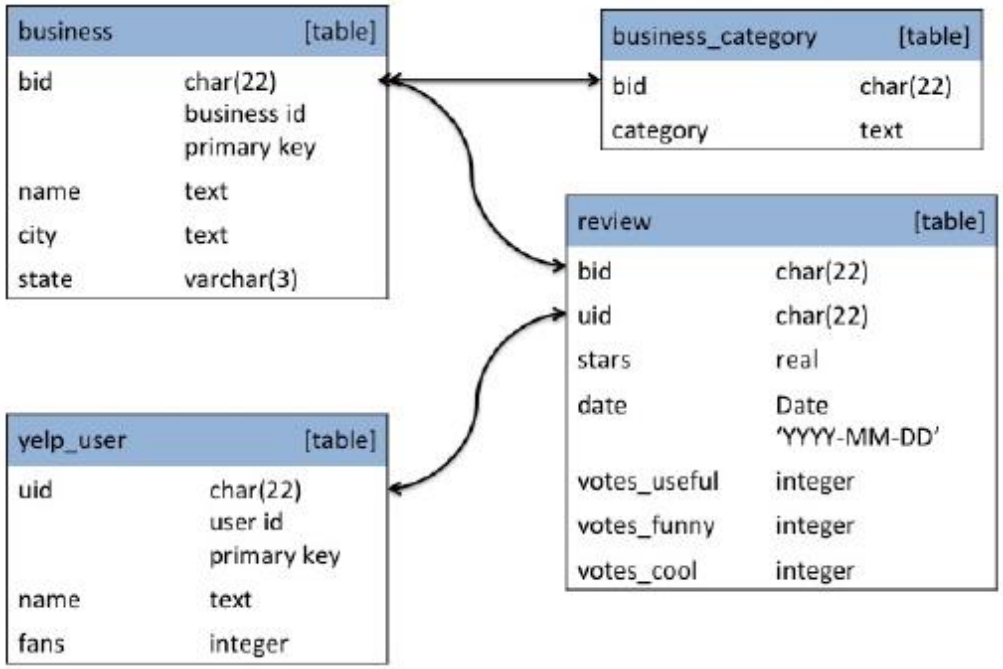
Ερωτήσεις;

Άσκηση: Αξιολογήσεις από το YELP



E1: Πλήθος διαφορετικών κατηγοριών επιχειρήσεων

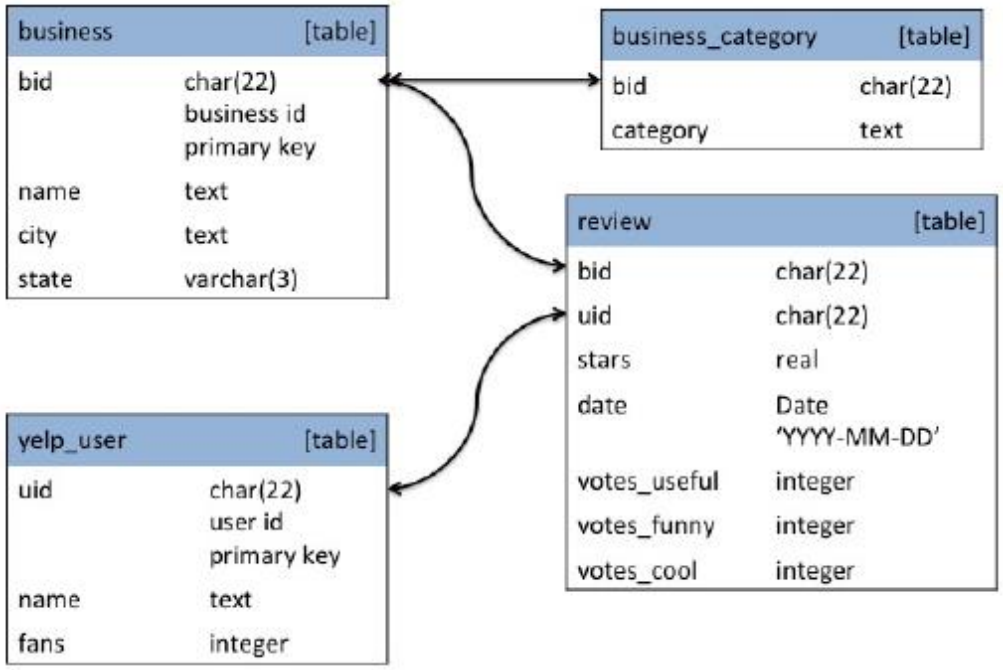
Άσκηση: Αξιολογήσεις από το YELP



E2: Τις πολιτείες των οποίων οι επιχειρήσεις έχουν λάβει συνολικά τις περισσότερες αξιολογήσεις

temp <- πολιτεία, #αξιολογήσεων

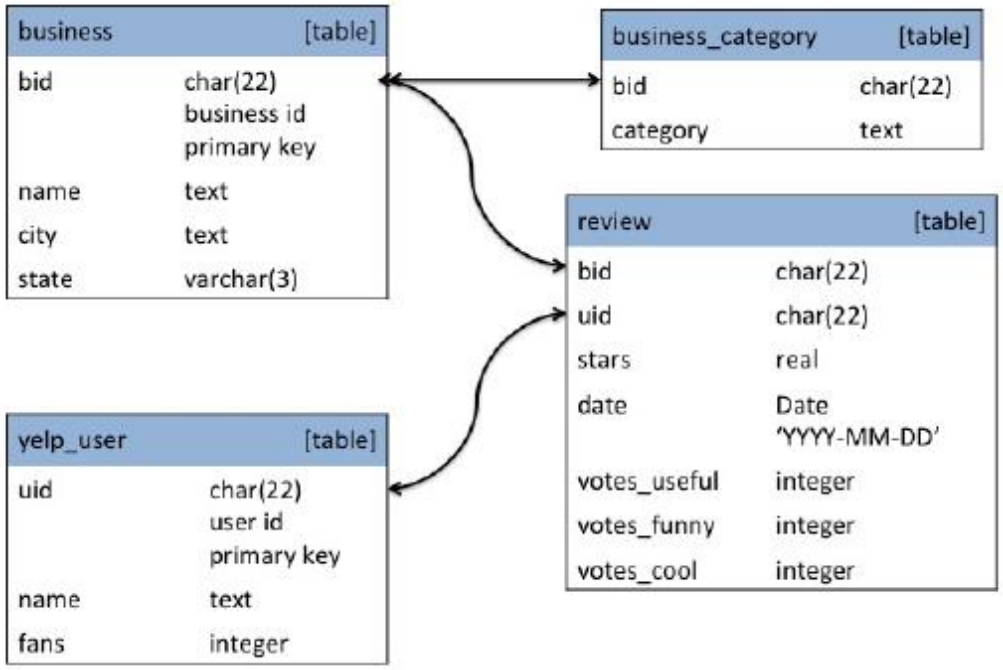
Άσκηση: Αξιολογήσεις από το YELP



E3: Τις επιχειρήσεις στην PA που έχουν τον όρο 'Coffee' στο όνομα τους αλλά δεν έχουν κατηγοριοποιηθεί ως «coffee place», δηλαδή ο όρος 'Coffee' δεν εμφανίζεται σε καμία από τις κατηγορίες στις οποίες ανήκουν. Δώστε το bid και το όνομα σε αύξουσα διάταξη του bid

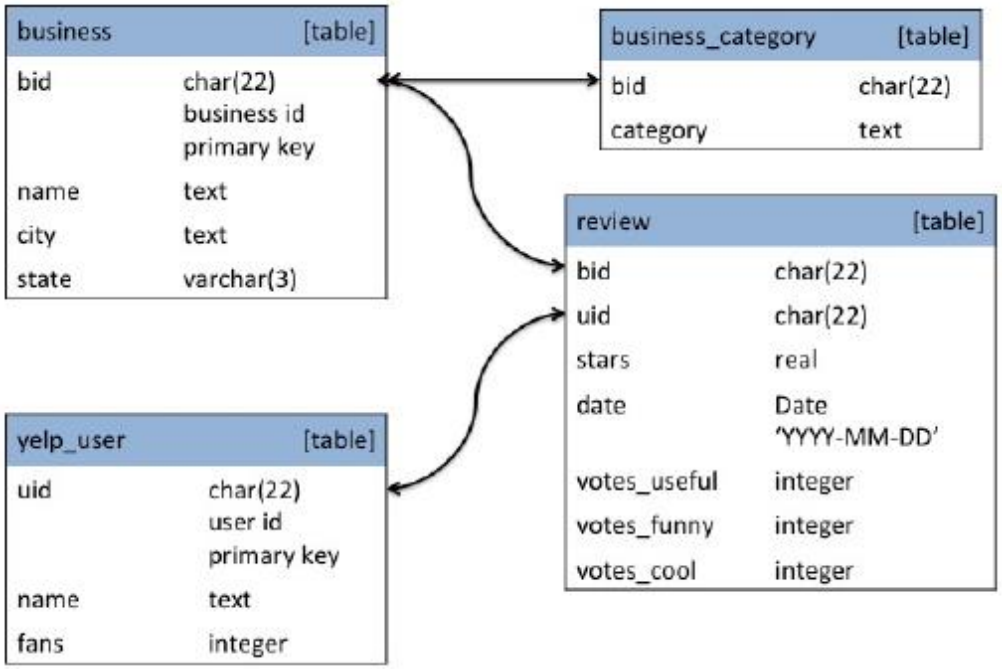
bid που έχει το Coffee και δεν ανήκουν (NOT IN) σε αυτές με Coffee στο category

Άσκηση: Αξιολογήσεις από το YELP



E3: Τις επιχειρήσεις στην PA που έχουν τον όρο 'Coffee' στο όνομα τους αλλά δεν έχουν κατηγοριοποιηθεί ως «coffee place», δηλαδή ο όρος 'Coffee' δεν εμφανίζεται σε καμία από τις κατηγορίες στις οποίες ανήκουν. Δώστε το bid και το όνομα σε αύξουσα διάταξη του bid

```
SELECT bid , name
FROM business
WHERE name LIKE `%Coffee%‘ AND state = `PA'
AND bid NOT IN (
SELECT bid
FROM business_category
WHERE category LIKE `%Coffee%'
)
ORDER BY bid ASC;
```



E4: Το πιο δημοφιλές bar σε κάθε πολιτεία (state)

Όπου πιο δημοφιλές αναφέρεται στην επιχείρηση με το μεγαλύτερο αριθμό reviews ανάμεσα στις επιχειρήσεις που μια από τις κατηγορίες τους είναι 'Bar'. Τυπώστε, για κάθε bar το bid, αριθμό review και πολιτεία σε αύξουσα διάταξη με το όνομα της πολιτείας και σε περίπτωση ισοβαθμίας σε αύξουσα με βάση το bid.

Temp1 <- bar, #review

Temp <-bar, #review, state

Ασκήσεις
(Θέματα Σεπτεμβρίου 2017)

Το παρακάτω σχεσιακό σχήμα μιας βάσης δεδομένων περιέχει πληροφορίες για αγώνες κολύμβησης.

ATHLETE(athlete_id, country, name, age)
EVENT(event_id, name)
EVENT_RESULT(event_id, athlete_id, result)

Ο πίνακας **ATHLETE** περιέχει πληροφορίες για τους αθλητές, συγκεκριμένα το μοναδικό id, τη χώρα, όνομα, και ηλικία του αθλητή.
Ο πίνακας **EVENT** περιέχει πληροφορίες για τα αγωνίσματα, συγκεκριμένα το μοναδικό id και το όνομα (πχ "100m sprint") του αγωνίσματος.
Ο πίνακας **EVENT_RESULT** περιέχει τους αθλητές που πήραν μετάλλια σε κάθε αγώνισμα, συγκεκριμένα το αγώνισμα (event_id), τον αθλητή (athlete_id) και το μετάλλιο (result) που αυτός πήρε. Το γνώρισμα result παίρνει τις τιμές, Gold, Silver και Bronze.

athlete_id	country	name	age
A1	U.S.A.	Michael Phelps	31
A2	U.S.A.	Justin Gatlin	34
A3	U.S.A.	Ryan Lochte	32
A4	Canada	Andre De Grasse	21
A5	Jamaica	Usain Bolt	30
A6	France	Christophe Lemaitre	26
A7	Japan	Masato Sakai	24
A8	Japan	Naito Ehara	60
A9	GBR	Duncan Scott	35
A10	GBR	James Guy	32

event_id	name
E1	100m Sprint
E2	200m Sprint
E3	200m Butterfly
E4	4x200 Freestyle Relay

event_id	athlete_id	result
E1	A5	Gold
E1	A2	Silver
E1	A4	Bronze
E2	A5	Gold
E2	A4	Silver
E3	A1	Gold
E3	A7	Silver
E3	A9	Bronze
E4	A1	Gold
E4	A3	Gold
E4	A7	Silver
E4	A8	Silver
E4	A9	Bronze
E4	A10	Bronze

ATHLETE

athlete_id	country	name	age
A1	U.S.A.	Michael Phelps	31
A2	U.S.A.	Justin Gatlin	34
A3	U.S.A.	Ryan Lochte	32
A4	Canada	Andre De Grasse	21
A5	Jamaica	Usain Bolt	30
A6	France	Christophe Lemaitre	26
A7	Japan	Masato Sakai	24
A8	Japan	Naito Ehara	60
A9	GBR	Duncan Scott	35
A10	GBR	James Guy	32

EVENT

event_id	name
E1	100m Sprint
E2	200m Sprint
E3	200m Butterfly
E4	4x200 Freestyle Relay

EVENT_RESULT

event_id	athlete_id	result
E1	A5	Gold
E1	A2	Silver
E1	A4	Bronze
E2	A5	Gold
E2	A4	Silver
E3	A1	Gold
E3	A7	Silver
E3	A9	Bronze
E4	A1	Gold
E4	A3	Gold
E4	A7	Silver
E4	A8	Silver
E4	A9	Bronze
E4	A10	Bronze

(α) Για καθένα από τα παρακάτω ερωτήματα εξηγείστε με απλά λόγια τι σημαίνουν και δώστε το αποτέλεσμα τους (σε μορφή πίνακα) όταν εκτελεστούν στο παρακάτω στιγμιότυπο.

(i) $\sigma_{age<25}$(ATLETE * EVENT_RESULT), όπου * η φυσική συνένωση

(ii) $\pi_{athlete\_id,\ event\_id}$(EVENT_RESULT) ÷ π_{event_id}($\sigma_{athlete\_id\ =\ 'A5'}$ (EVENT_RESULT))

(iii) {t.name | EVENT(t) AND (∃r (EVENT_RESULT(r) AND r.athlete_id = 'A4' AND t.event_id = r.event_id))}

ATHLETE(athlete_id, country, name, age)
EVENT(event_id, name)
EVENT_RESULT(event_id, athlete_id, result)

(β) Δώστε ερωτήσεις σε σχεσιακή άλγεβρα που να έχουν ως αποτέλεσμα:
(i) το id των αθλητών που έχουν κερδίσει μόνο χρυσά (gold) μετάλλια
(ii) το id των Αμερικάνων αθλητών που πήραν μετάλλιο στο αγώνισμα με όνομα "100m Sprint"

ATHLETE(athlete_id, country, name, age)
EVENT(event_id, name)
EVENT_RESULT(event_id, athlete_id, result)

(γ) Δώστε ερωτήσεις σε SQL που να έχουν ως αποτέλεσμα:
(i) για κάθε αθλητή τον αριθμό των μεταλλίων που κέρδισε (ζεύγη: id-αθλητή, αριθμός μεταλλίων) σε φθίνουσα διάταξη βάσει του αριθμού μεταλλίων και σε περίπτωση ισοβαθμίας με αύξουσα διάταξη με βάση το id, αγνοείστε όσους αθλητές δεν πήραν μετάλλια
(ii) τροποποιείστε την ερώτηση γ(i) ώστε να περιλαμβάνονται στην απάντηση και οι αθλητές που δεν πήραν μετάλλια (να εμφανίζονται με αριθμό μεταλλίων 0).
(iii) το id των αθλητών που έχουν κερδίσει μόνο χρυσά (gold) μετάλλια χρησιμοποιώντας in/not in.
(iv) τις χώρες που έχουν πάρει τουλάχιστον 5 μετάλλια (ζεύγη: χώρα, αριθμός μεταλλίων) σε φθίνουσα διάταξη βάσει του αριθμού μεταλλίων (υποθέστε ότι τα αγωνίσματα είναι ατομικά).

Λύση 1ου quiz
(Ακ. Έτος 2020-2021)

Ερώτηση 1

```
CREATE TABLE CUSTOMER (
CustomerID INTEGER PRIMARY KEY,
Name VARCHAR NOT NULL
);
CREATE TABLE BASKET (
BasketID    INTEGER PRIMARY KEY,
CustomerID INTEGER REFERENCES CUSTOMER (CustomerID) ON DELETE CASCADE
UNIQUE
);
```

Ο πίνακας CUSTOMER έχει τις εξής πλειάδες:

CustomerID	Name
2	Smith
4	Brown
3	Andrews
7	Jordan
10	Pappas

Ποια από τα ακόλουθα στιγμιότυπα δεν μπορεί να είναι έγκυρα στιγμιότυπο του πίνακα BASKET;

A

BasketID	CustomerID
1	2
2	3
3	2

B

BasketID	CustomerID
1	2
4	4
6	3

C

BasketID	CustomerID
1	2
2	4
3	3

D

BasketID	CustomerID
1	2
2	4
3	6

Για τη βάση δεδομένων των εργαζομένων μιας επιχείρησης δημιουργήσαμε τους πίνακες DEPARTMENT (τμήμα), EMPLOYEE (εργαζόμενος) και WORKS (δουλεύει):

Ερώτηση 2

```
CREATE TABLE DEPARTMENT (
DepartmentID      INTEGER PRIMARY KEY,
Name   VARCHAR  UNIQUE NOT NULL
);
CREATE TABLE EMPLOEE (
EmployeeID INTEGER PRIMARY KEY,
Name VARCHAR NOT NULL
);
CREATE TABLE WORKS (
EmployeeID INTEGER REFERENCES Employees (EmployeeID) ON DELETE CASCADE,
DepartmentID INTEGER REFERENCES Departments (DepartmentID) ON DELETE CASCADE,
PRIMARY KEY (EmployeeID, DepartmentID)
);
```

Η βάση υλοποιεί δύο τύπους οντοτήτων (Ο1: EMPLOYEE, Ο2: DEPARTMENT) και τη μεταξύ τους σχέση (WORKS). Ποιον περιορισμό πληθικότητας υλοποιεί το παραπάνω σχήμα;

Επιλέξτε ένα ή περισσότερα:

A. 1:1 (ένα τμήμα μπορεί να έχει μόνον έναν εργαζόμενο και ένας εργαζόμενος δουλεύει μόνο σε ένα τμήμα).
B. N:1 (ένα τμήμα μπορεί να έχει πολλούς εργαζόμενους και ένας εργαζόμενος δουλεύει μόνο σε ένα τμήμα).
C. 1:N (ένας εργαζόμενος μπορεί να δουλεύει σε πολλά τμήματα και ένα τμήμα έχει μόνο έναν εργαζόμενο).
D. M:N (ένας εργαζόμενος μπορεί να ανήκει σε πολλά τμήματα και ένα τμήμα να έχει πολλούς εργαζόμενους).

Σε κάποιες χώρες ομιλούνται περισσότερες από μία γλώσσες από τους κατοίκους της χώρας. Ο παρακάτω πίνακας δημιουργήθηκε για να περιγράψει τις γλωσσικές διαφοροποιήσεις σε κάθε χώρα της υφηλίου.

```
CREATE TABLE LANGUAGE-SPOKEN (
CountryID   INTEGER,
LanguageID INTEGER,
Province    VARCHAR,
PRIMARY KEY (CountryID, Province)
);
```

Ποια από τις παρακάτω υποθέσεις υλοποιεί ο πίνακας;
Επιλέξτε ένα ή περισσότερα:
A. Σε κάθε επαρχία (province) μιας χώρας ομιλείται μία γλώσσα.
B. Σε κάθε επαρχία (province) μιας χώρας ομιλείται μία ή περισσότερες γλώσσες.
C. Κάθε γλώσσα ομιλείτε το πολύ σε μια επαρχία (province) μιας χώρας.
D. Σε κάθε επαρχία (province) μιας χώρας ομιλείται η επίσημη γλώσσα της χώρας.

Ερώτηση 4

Μια βάση δεδομένων για ένα ηλεκτρονικό κατάστημα περιλαμβάνει τους πίνακες BASKET και PRODUCT. Κάθε εγγραφή του πίνακα PRODUCT σχετίζεται με μία ή καμία εγγραφή του πίνακα BASKET. Έστω ότι ορίζουμε τον πίνακα PRODUCT ως εξής:

```
CREATE TABLE PRODUCT (
ProductID   INTEGER PRIMARY KEY,
BasketID INTEGER REFERENCES Baskets (BasketID) ON DELETE SET NULL
);
```

Ένα στιγμιότυπο αυτού του πίνακα είναι το εξής:

ProductID	BasketID
1	2
2	1

Πληκτρολογούμε την εντολή: DELETE FROM BASKET WHERE BasketID = 2;

Τι από τα ακόλουθα ισχύει για τον πίνακα PRODUCT;

Επιλέξτε ένα ή περισσότερα:

A. Η πλειάδα με ProductID = 1 θα διαγραφεί.
B. Η πλειάδα με ProductID = 2 θα διαγραφεί.
C. Η πλειάδα με ProductID = 1 θα τροποποιηθεί.
D. Η πλειάδα με ProductID = 2 θα τροποποιηθεί.

Ερώτηση 5

Η βάση δεδομένων μιας κλινικής περιλαμβάνει τις οντότητες PATIENT, MEDICINE, CATEGORY, και υλοποιεί τη σχέση PRESCRIPTION. Ο πίνακας CATEGORY περιλαμβάνει κατηγορίες φαρμάκων, ο πίνακας MEDICINE περιλαμβάνει φάρμακα όλων των κατηγοριών και ο πίνακας PATIENT περιλαμβάνει τους ασθενείς. Η σχέση PRESCRIPTION περιγράφει τα φάρμακα που έχουν συνταγογραφηθεί για κάθε ασθενή που υπάρχει στον πίνακα PATIENT. Φάρμακα που διαγράφονται από τον πίνακα MEDICINE δεν θα πρέπει περιλαμβάνονται στη συνταγή ενός ασθενούς. Οι πίνακες CATEGORY, MEDICINE, PATIENT υλοποιούνται ως εξής:

CREATE TABLE **CATEGORY** (CatID INTEGER PRIMARY KEY, Name VARCHAR NOT NULL);

CREATE TABLE **MEDICINE** (MedID INTEGER PRIMARY KEY, Name VARCHAR NOT NULL, CatID INTEGER REFERENCES CATEGORY (CatID) ON DELETE SET NULL,);

CREATE TABLE **PATIENT** (PatientID INTEGER PRIMARY KEY, Name VARCHAR NOT NULL);

Ποια υλοποίηση του πίνακα PRESCRIPTION εκφράζει τη σχέση που περιγράψαμε;

Επιλέξτε ένα ή περισσότερα:

A. CREATE TABLE PRESCRIPTION(
PrescrID INTEGER PRIMARY KEY,
PatientID INTEGER REFERENCES PATIENT (PatientID) ON DELETE CASCADE,
MedID INTEGER REFERENCES MEDICINE (MedID) ON DELETE CASCADE,
CatID INTEGER REFERENCES CATEGORY (CatID) ON DELETE SET NULL
);

B. CREATE TABLE PRESCRIPTION(
PrescrID INTEGER PRIMARY KEY,
PatientID INTEGER REFERENCES PATIENT (PatientID) ON DELETE CASCADE,
MedID INTEGER REFERENCES MEDICINE (MedID) ON DELETE SET NULL,
CatID INTEGER REFERENCES CATEGORY (CatID) ON DELETE SET NULL
);

C. CREATE TABLE PRESCRIPTION(
PrescrID INTEGER PRIMARY KEY,
PatientID INTEGER REFERENCES PATIENT (PatientID) ON DELETE CASCADE,
MedID INTEGER REFERENCES MEDICINE (MedID) ON DELETE CASCADE,
CatID INTEGER REFERENCES CATEGORY (CatID) ON DELETE CASCADE
);

Έστω μια βάση δεδομένων με τρεις πίνακες R, S και Q ορισμένους ως:
CREATE **TABLE R** (A INT PRIMARY KEY, B INT);
CREATE **TABLE S** (C INT PRIMARY KEY, D INT, FOREIGN KEY(D) REFERENCES R(A) ON DELETE CASCADE ON UPDATE CASCADE);
CREATE **TABLE Q** (E INT PRIMARY KEY, F INT, FOREIGN KEY(F) REFERENCES S(C) ON DELETE CASCADE ON UPDATE CASCADE);
S(D)
Στο τρέχον στιγμιότυπο της βάσης δεδομένων, οι πίνακες είναι όπως παρακάτω:

R	
A	B
1	2
2	3
3	1

S	
C	D
3	1
5	1
6	2

Q	
E	F
6	1
5	1
3	2
7	2

Επιλέξτε ένα ή περισσότερα:
A. Η εντολή DROP TABLE S δε γίνεται δεκτή.
B. Μετά την εκτέλεση της εντολής DELETE FROM R WHERE A = 1, ο πίνακας Q θα έχει 2 πλειάδες.
C. Μετά την εκτέλεση της εντολής DELETE FROM R WHERE A = 1, ο πίνακας Q θα έχει 4 πλειάδες.
D. Η εντολή UPDATE S SET D = 3 WHERE D = 1 δε γίνεται δεκτή.
E. Μετά την εκτέλεση της εντολής UPDATE S SET D = 3 WHERE D = 1, ο πίνακας Q θα έχει 4 πλειάδες.
F. Η εντολή UPDATE Q SET E = 6 WHERE F = 2 δε γίνεται δεκτή.
G. Μετά την εκτέλεση της εντολής UPDATE Q SET E = 6 WHERE F = 2, ο πίνακας Q θα έχει 4 πλειάδες.