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 (εκτός ύλης για Ακαδημαϊκό Έτος 2023-2024)

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)

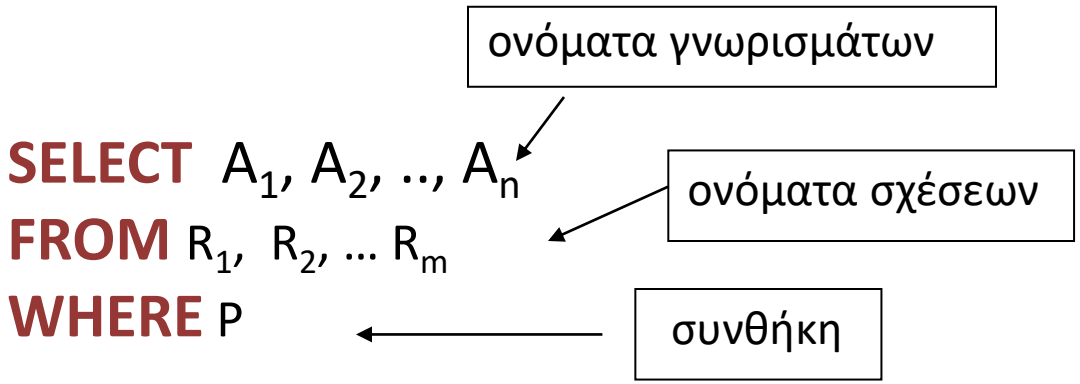
Προδιαγραφές ασφάλειας - χρήστες και δικαιώματα.

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

Παράδειγμα

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

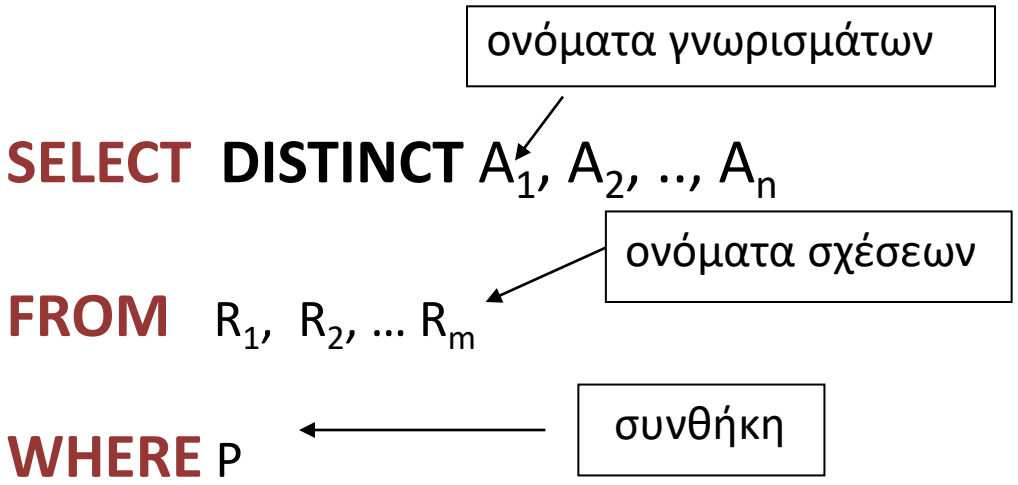
Παράδειγμα φυσικής συνένωσης:
Τους ηθοποιούς που παίζουν σε ασπρόμαυρες ταινίες

```
SELECT DISTINCT Name
FROM  Plays [INNER] JOIN  Movie ON
    Plays.Title = Movie.Title AND Plays.Year = Movie.Year
WHERE  Type = 'Ασπρόμαυρη'
```

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.

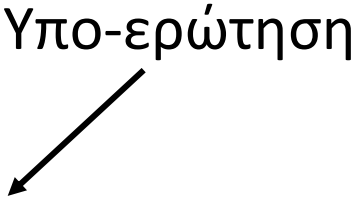
Σύνταξη

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 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 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 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 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

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

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 = 'Έγχρωμη');

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

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

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 R WHERE B IN (SELECT D FROM S);

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

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

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?

Κάποιες online πηγές

W3Schools SQL Tutorial: A comprehensive tutorial that covers the basics of SQL.

SQLBolt: Provides interactive lessons and exercises to teach you SQL.

SQLZoo: A collection of SQL tutorials with various challenges to test your knowledge.

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 INNER JOIN PIZZA
      ON PIZZA.Ingredient  = LIKES.Ingredient);
```

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

Φυσική Συνένωση (natural join)

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

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

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

Τι επιστρέφει το παρακάτω?

```
SELECT DISTINCT LIKES. Student
FROM LIKES LEFT OUTER JOIN PIZZA ON LIKES.Ingredient = PIZZA.Ingredient
WHERE PIZZA.Name IS NULL;
```

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

Παράδειγμα

PIZZA(Name, Ingredient)

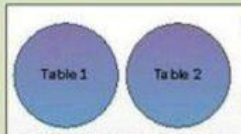
LIKES(Student, Ingredient)

Τι επιστρέφει το παρακάτω?

```
SELECT DISTINCT PIZZA.Name
FROM LIKES RIGHT OUTER JOIN PIZZA ON LIKES.Ingredient = PIZZA.Ingredient
WHERE LIKES.Student IS NULL;
```

MySQL JOIN Types

Created by Steve Stedman



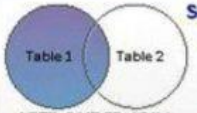

```
SELECT *
  FROM Table1;

SELECT *
  FROM Table2;
```

SELECT from two tables


```
SELECT *
  FROM Table1 t1
INNER JOIN Table2 t2
  ON t1.fk = t2.id;
```

INNER JOIN

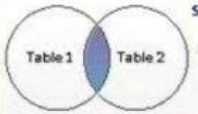



```
SELECT *
  FROM Table1 t1
LEFT OUTER JOIN Table2 t2
  ON t1.fk = t2.id;
```

LEFT OUTER JOIN

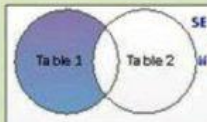

```
SELECT *
  FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
  ON t1.fk = t2.id;
```

RIGHT OUTER JOIN



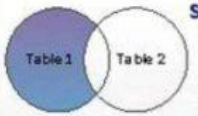

```
SELECT *
  FROM Table1 t1
WHERE EXISTS (SELECT 1
                FROM Table2 t2
               WHERE t1.fk = t2.id
              );
```

SEMI JOIN – Similar to INNER JOIN, with less duplication.



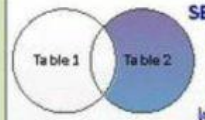

```
SELECT *
  FROM Table1 t1
WHERE NOT EXISTS (SELECT 1
                    FROM Table2 t2
                   WHERE t1.fk = t2.id
                  );
```

ANTI SEMI JOIN



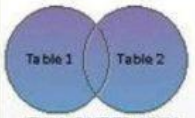

```
SELECT *
  FROM Table1 t1
LEFT OUTER JOIN Table2 t2
  ON t1.fk = t2.id
WHERE t2.id is null;
```

LEFT OUTER JOIN with exclusion



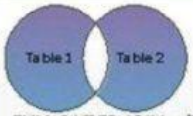

```
SELECT *
  FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
  ON t1.fk = t2.id
WHERE t1.fk is null;
```

RIGHT OUTER JOIN with exclusion



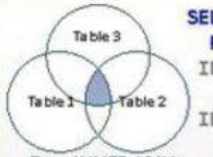

```
SELECT * FROM Table1 t1
  LEFT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
UNION
SELECT * FROM Table1 t1
  RIGHT OUTER JOIN Table2 t2
    ON t1.fk = t2.id;
```

FULL OUTER JOIN



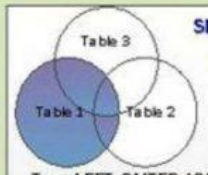

```
SELECT * FROM Table1 t1
  LEFT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t2.id IS NOT NULL
UNION
SELECT * FROM Table1 t1
  RIGHT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t1.ID IS NOT NULL;
```

FULL OUTER JOIN with exclusion



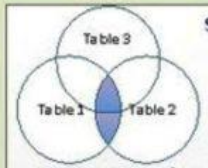

```
SELECT *
  FROM Table1 t1
INNER JOIN Table2 t2
  ON t1.fk = t2.id
INNER JOIN Table3 t3
  ON t1.fk_table3 = t3.id;
```

Two INNER JOINs




```
SELECT *
  FROM Table1 t1
LEFT OUTER JOIN Table2 t2
  ON t1.fk = t2.id
LEFT OUTER JOIN Table3 t3
  ON t1.fk_table3 = t3.id;
```

Two LEFT OUTER JOINS




```
SELECT *
  FROM Table1 t1
INNER JOIN Table2 t2
  ON t1.fk = t2.id
LEFT OUTER JOIN Table3 t3
  ON t1.fk_table3 = t3.id;
```

INNER JOIN and a LEFT OUTER JOIN

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*

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 τύπο ταινίας

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

Στο SELECT και η τιμή του γνωρίσματος του GROUP BY

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

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;
```

Παράδειγμα

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

SERVES(Place, Name)

Τι επιστρέφουν τα παρακάτω?

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

```
SELECT COUNT(DISTINCT Ingredient)
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: 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(*)
1  FROM Plays, Actor
2  WHERE  Plays.Name = Actor.Name   AND  Year-of-Birth> 1987
3  GROUP BY Actor.Name
4  HAVING COUNT(*) >= 5;
```

5

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
GROUP BY A
HAVING MAX(B) > 2;
```

```
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

- Οι συναρτήσεις μας δίνουν την τιμή όχι που εμφανίζεται αυτή

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

Η τιμή του B είναι μεγαλύτερη από τη *μέση τιμή*?

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 $A_{j1}, A_{j2}, \ldots, A_{jk}$

- Τα αποτελέσματα των SQL ερωτημάτων ΔΕΝ αποθηκεύονται
- Σε πιο δύσκολα ερωτήματα μπορεί να χρειαστεί να έχουμε ενδιάμεσα αποτελέσματα, πως;

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;
```

Τι κάνει το παραπάνω ερώτημα;

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

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

Σειρά εκτέλεσης:

1. FROM
2. WHERE
3. GROYP BY
4. HAVING
5. ORDER BY
6. LIMIT
7. SELECT

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

SERVES(Place, Name)

1. Πόσα μαγαζιά υπάρχουν στη βάση δεδομένων.

2. Τα ονόματα των μαγαζιών που σερβίρουν 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

Επιπρόσθετα, για τις πίτσες που δεν έχουν κανένα τέτοιο συστατικό να εμφανίζεται ο αριθμός 0

PIZZA(Name, Ingredient)

LIKES(Student, Ingredient)

SERVES(Place, Name)

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)

Τους φοιτητές που τους αρέσουν όλα τα συστατικά που αρέσουν στο Δημήτρη

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>

Let's Ask the Experts: Insights into Student Misconceptions about SQL

Daphne Miedema
d.e.miedema@tue.nl

Efthimia Aivaloglou
e.aivaloglou@liacs.leidenuniv.nl

George Fletcher
g.h.l.fletcher@tue.nl

The Structured Query Language (SQL) has been introduced in 1970, and quickly became the standard query language for Relational Databases. It is ubiquitous in both practice and education. Various researchers have published works on the errors that novices make while writing queries. However, until recently, not much attention was paid to the underlying reasons why novices made so many mistakes. In a previous paper, we explored these underlying reasons, also called misconceptions, from the student perspective. We gave students query formulation problems and asked them to solve those while explaining to us what they were thinking. We followed this up by exploring the perspectives of SQL experts -researchers, practitioners and educators-. Their expertise on errors and solutions gave us new insights into other misconceptions that may lead to the same error. The result are 11 lists of misconceptions that novices might hold. Below, we present the top-ranked expert hypothesis and juxtapose it with the student hypothesis.

1. Using ≠ instead of != or <>

```
SELECT *
FROM store
WHERE city ≠ "Eindhoven";
```

Expert: Students are familiar with ≠ and find it more intuitive than <> and !=.

Student: Interference of previous course knowledge from either Mathematics or Relational Algebra.

2. AVG in WHERE instead of HAVING

```
GROUP BY t.sID
FROM transaction as t
WHERE AVG(t.quantity) > 4;
```

Expert: Students associate the application of a filter with WHERE.

Student: Misunderstanding SQL syntax and its internal (in)consistency.

3. JOIN in all tables that you need the primary keys of

```
SELECT c.cID, t.date,
    t.quantity
FROM customer as c,
transaction as t,
product as p
WHERE t.pID = p.pID
AND c.cID = t . cID
AND p.pName = "Apples"
```

Expert and Student: Students have created a template query/pattern and apply it without understanding it.

4. Missing second table for self-join

```
INSERT cID FROM customers
WHERE street = street
AND city <> city;
```

Expert: Students do not think of a second access as a separate instance of a table.

Student: Misunderstanding the scope of elements in a query.

5. Lack of understanding of when to apply GROUP BY

```
SELECT city, street,
   sName, sID
FROM store
GROUP BY city
ORDER BY city ASC;
```

Expert: Students do not understand the implication of grouping

Student: Misunderstanding SQL syntax and its internal (in)consistency.

Delphi protocol

Hypotheses types

- Previous course knowledge
- Generalisations
- Incorrect or incomplete mental model
- Language related

6. Missing parts of keywords

```
SELECT *
FROM store AS s
ORDER s.city ASC
```

Expert: Students forget the correct syntax because most other SQL keywords are single-valued.

Student: Remembering only the core part of a keyword as it captures the main meaning.

For more information about student misconceptions in SQL, find our open-access paper here:

7. Extra condition when not required

```
SELECT c1.cID, c2.cID
FROM customer c1, customer c2
WHERE c1.cID <> c2.cID
AND c1.street = c2.street
AND c1.city <> c2.city ;
```

Expert: Students do not realise that being a different customer follows from the different cities.

Student: Lacking knowledge on primary keys.

8. Ambiguous column name because of missing alias

```
SELECT unit_price
FROM inventory
INNER JOIN product
ON pID = pID
WHERE pName = "Bananas"
```

Expert: Student knew which pID they meant and feel it is self-explanatory.

Student: Lack of knowledge or experience.

9. Using comma instead of AND in WHERE clause

```
SELECT p.pName
FROM inventory as I,
  product as p
WHERE
([subquery]) < i.unit_price,
p.pID= i.pID;
```

Expert: Student was sloppy in writing the query.

Student: Misunderstanding SQL syntax and its internal (in)consistency.

10. Thinking DISTINCT will take the first item of a list

"If I would have used a DISTINCT before, on the unit_price, then you'd have taken the first unit_price and compared it with the unit_price of the primary query. [...] If I used DISTINCT over this C.unit_price, I'd have given only one unit_price and probably that would have been the first one, because that is what it will find first. And then it would just compare [the price] with that one."

Expert: Students mix the semantics of 'a single' and 'the first'.

Student: Believing DISTINCT will take the first item of a list (because of being presented with very selective examples of DISTINCT queries.)

11. Lack of understanding subqueries

```
SELECT t.cID, t.date,
t.quantity
FROM transaction as t
WHERE (
   SELECT pName
   FROM product
   WHERE pID = t.pID
   AND pName = "Apples")
```

Expert: Students prefer a loop-style query, as you would do in programming.

Student: Lack of knowledge or experience.

Conclusion

Research Question 1: *How do experts interpret the underlying causes of student errors in SQL?*
The varying backgrounds of the experts meant that they approached the problem from different angles. We found up to 8 different hypotheses per error, some were deemed more likely than others. Sometimes all hypotheses were closely related, but the experts typically considered a wide spectrum of underlying causes.

Research Question 2: *What are the differences in the perspectives of students and experts on errors in SQL?*
Although sometimes there was a difference in focus, overall the students misconception hypotheses were ranked highly by the experts too. For example, for error 2, the wordings are different but the core of the hypotheses for this error is the same. The only error where the students and experts were not in agreement was error 6.

Data Science - Data Prep with SQL - Quick Reference

DATASET PROFILING

Volume	SELECT **COUNT**(*) FROM t;
Velocity	SELECT t.date1, COUNT(*) FROM t **GROUP by** t.date1 ORDER BY t.date1 desc;
Attribute Selection	SELECT attr1, attr2, attr3, attr4 FROM t;
Incomplete Records	SELECT * FROM t WHERE t.attr1 **IS NULL** AND t.attr2 IS NULL;

VALIDATE ATTRIBUTES

Domain	SELECT **DISTINCT**(attr1) FROM t;
Missing Values	SELECT * FROM t WHERE t.attr1 **IS NULL**;
Range	SELECT **MIN**(attr1), **MAX**(attr1), **AVG**(attr1) FROM t;
Data Type	SELECT * FROM information_schema.**columns** WHERE table_name = 't';
Outliers (95% confidence)	WITH dev_cte AS (SELECT **STDDEV**(attr1) sdev FROM t) SELECT attr1, attr2 FROM t CROSS JOIN dev_cte c WHERE t.attr1 > c.sdev * 2;
Distribution	SELECT attr1, **WIDTH_BUCKET**(attr1,100,500,5) FROM t;

STANDARDIZE ATTRIBUTES

Data Types	SELECT **CAST**(attr1 AS DATE), CAST(attr2 AS INT) FROM t;
Patterns	SELECT **CASE** WHEN attr1 = …, **REPLACE**(attr2,'Street','St') FROM t;
Formatting	SELECT **UPPER**(attr1), **REPLACE**(attr2,'-','') FROM t;
Scaling	SELECT attr1, attr2/(**MAX**(attr2) OVER (**PARTITION** BY attr1)) FROM t;

CREATE INTERFACE

Create view	CREATE **VIEW** AS SELECT…

CLEAN ATTRIBUTES

Outliers (Quantitative)	SELECT **CASE** WHEN attr1 < 0 THEN 0 WHEN attr1 > 1000 THEN 1000 ELSE attr1 END as attr1 FROM t;
Missing Values (At Random)	SELECT **COALESCE**(attr1,AVG(attr1) OVER ()), **COALESCE** (attr1,'Unknown') FROM t;
Missing Values (Not at Random)	SELECT **COALESCE**(attr1,0) FROM t;
Incorrect Values	SELECT **REPLACE**(attr1,'bad','good') FROM t;

DERIVE ATTRIBUTES

Buckets\Binning	SELECT attr1, **CASE** WHEN attr1 <= 50 THEN 'bin1' WHEN attr1 > 50 THEN 'bin2' ELSE 'bin3' END as attr1_bin FROM t;
Date Parts	SELECT **DAYOFMONTH**(date1), **MONTHOFYEAER**(date1) FROM t;
Date Difference	SELECT **DATEDIFF**(date1,date2) FROM t;
Last Period	SELECT **DATEADD**(year,-1,date1) FROM t;
Dummy Encoding (One Hot)	SELECT attr1, **CASE** WHEN attr1 = 'Male' THEN 1 ELSE 0 as male_gender FROM t;

COMBINE DATASETS

Join Horizontally (Full Match)	SELECT t1.attr1, t2.attr2 FROM t1 **INNER JOIN** t2 ON t1.ID = t2.ID;
Join Horizontally (Optional Match)	SELECT t1.attr1, t2.attr2 FROM t1 **LEFT JOIN** t2 ON t1.ID = t2.ID;
Union Vertically (Deduplicate)	SELECT attr1, attr2 FROM t1 **UNION** SELECT attr1, attr2 FROM t2
Union Vertically (No Deduplicate)	SELECT attr1, attr2 FROM t1 **UNION ALL** SELECT attr1, attr2 FROM t2

SPLIT DATASETS

Simple Filter	SELECT attr1, attr2 FROM t **WHERE** attr1 IS NOT NULL;
Filter Based on Aggregation	SELECT attr1, SUM(attr2) FROM t GROUP BY attr1 **HAVING** SUM(attr2) > 10;
Sampling (Random)	SELECT attr1, **ROW_NUMBER**() OVER (ORDER BY RANDOM()) as random FROM t;
Sampling (Non-Random)	SELECT attr1, **NTILE**(4) OVER (ORDER BY date()) as quartile FROM t;

Ερωτήσεις;