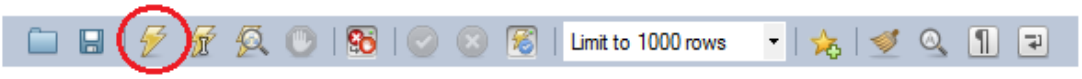
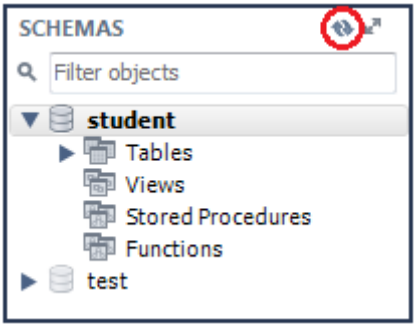
Εργαστηριακό μάθημα 3
-
Υλοποίηση Σύνθετων SQL ερωτημάτων στη βάση SongsDB

Δημιουργία της βάσης SongsDB

1. Ανοίξτε την εφαρμογή **MySQL Workbench** και συνδεθείτε στη βάση (όπως στο πρώτο εργαστήριο).
2. Αν το schema student υπάρχει ήδη διαγράψτε το. Κάνετε δεξί κλικ πάνω του, επιλέξτε "**Drop schema…**" και στο μενού που θα εμφανιστεί επιλέξτε "**Drop now**".
3. Επιλέξτε «**File -> Open SQL Script…**» για να ανοίξετε το αρχείο "***Lab2Dump.sql***" και στη συνέχεια πιέστε το κουμπί **«Execute»** (βρίσκεται στην γραμμή εργαλείων). Εναλλακτικά, μπορείτε να εκτελέσετε το script επιλέγοντας «**Query -> Execute (All or Selection)**» ή πιέζοντας Ctrl+Shift+Enter.



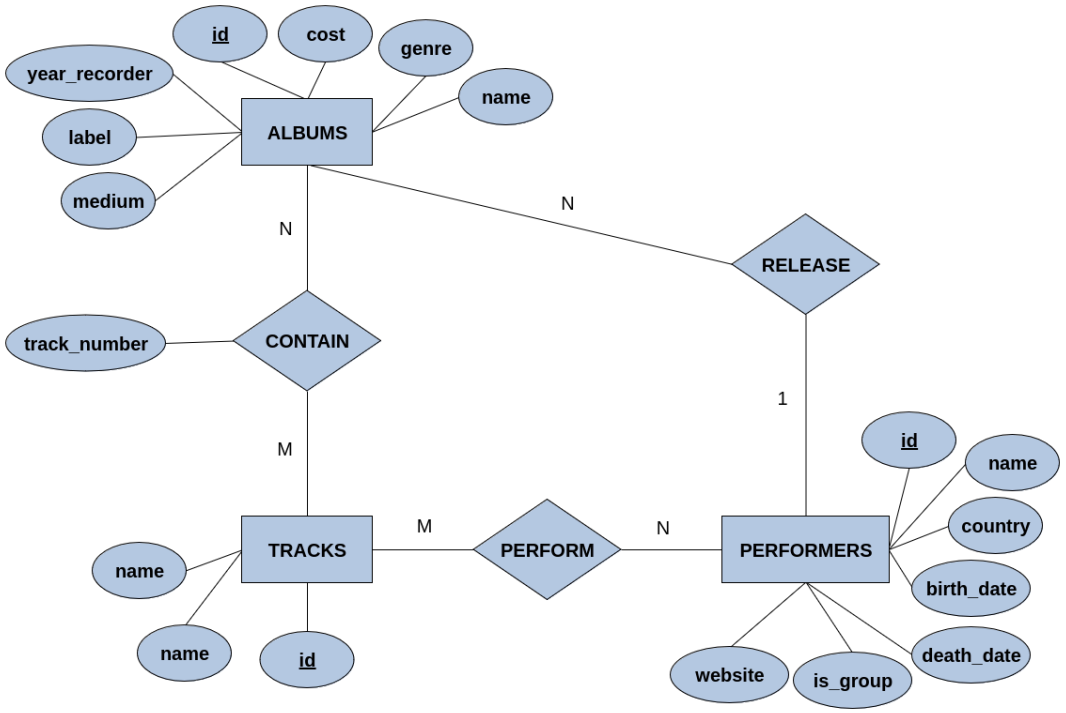
4. Ελέγξτε στο **«SCHEMAS»** (στα αριστερά του GUI) αν κάτω από το student έχουν όντως δημιουργηθεί όλοι οι πίνακες με τα ζητούμενα κλειδιά και δεδομένα. Θα χρειαστεί να κάνετε "refresh" τα SCHEMAS για να εμφανιστεί το σχήμα της ΒΔ "student".



5. Επιλέξτε «**File -> New Query Tab**» (εναλλακτικά μπορείτε να πατήσετε το κουμπί «**New Query Tab**» στην γραμμή εργαλείων Standard ή τον συνδυασμό πλήκτρων Ctrl+N).

9ο Εξάμηνο

Απλοποιημένο διάγραμμα Ο/Σ της βάσης SongsDB



SQL Ερωτήματα προς Υλοποίηση

1. Να βρεθούν ο τίτλος, το είδος και ο καλλιτέχνης όλων των albums που είναι καταχωρημένα στη βάση. Μετονομάστε τις στήλες στο αποτέλεσμα του ερωτήματος σε Album, Genre, Artist.

```
SELECT albums.name AS Album, genre AS Genre, performers.name AS Artist FROM
albums JOIN performers ON albums.performer_id = performers.id
```

Σημειώστε πως μπορεί εναλλακτικά να χρησιμοποιηθεί το **INNER JOIN** αντί του **JOIN**

Χρησιμοποιούμε τη JOIN για να συνδυάσουμε δεδομένα που βρίσκονται σε διαφορετικούς πίνακες. Εδώ, τα άλμπουμ και οι καλλιτέχνες έχουν τα δικά τους μοναδικά δεδομένα σε ξεχωριστούς πίνακες. Θέλουμε να δούμε τα ονόματα των καλλιτεχνών των άλμπουμ, αλλά στα άλμπουμ το μόνο που έχουμε είναι το performer_id του καλλιτέχνη. Χρησιμοποιώντας το performer_id, μπορούμε να "τραβήξουμε" το όνομά του από τον αντίστοιχο πίνακα.

2. Να βρεθούν όλα τα albums (τίτλος και είδος) που είναι καταχωρημένα στη βάση και ο αντίστοιχος καλλιτέχνης. Να εμφανίσετε επιπλέον τα albums που δεν αντιστοιχούν σε κάποιο καλλιτέχνη.

```
SELECT albums.name AS Album, genre AS Genre, performers.name AS Artist
FROM albums
  LEFT JOIN performers ON albums.performer_id = performers.id
```

Σημειώστε πως μπορεί εναλλακτικά να χρησιμοποιηθεί το **LEFT OUTER JOIN** *αντί του* **LEFT JOIN**

Το LEFT JOIN μας επιστρέφει όλα τα άλμπουμ από τον πίνακα albums, ακόμη κι αν δεν υπάρχει αντίστοιχος καλλιτέχνης στον πίνακα performers. Αν ένα άλμπουμ δεν αντιστοιχεί σε κάποιον καλλιτέχνη (δηλαδή αν το performer_id του άλμπουμ δεν ταιριάζει με κανένα id στον πίνακα performers), τότε στα αποτελέσματα θα εμφανιστεί κανονικά το άλμπουμ, αλλά η στήλη Artist θα είναι κενή (null).

3. Να βρεθούν όλα τα albums (τίτλος και είδος) που είναι καταχωρημένα στη βάση και ο αντίστοιχος καλλιτέχνης. Να εμφανίσετε επιπλέον τους καλλιτέχνες που δεν έχουν κάποιο album.

```
SELECT albums.name AS Album, genre AS Genre, performers.name AS Artist
FROM albums
  RIGHT JOIN performers ON albums.performer_id = performers.id
```

Σημειώστε πως μπορεί εναλλακτικά να χρησιμοποιηθεί το **RIGHT OUTER JOIN** *αντί του* **RIGHT JOIN**

Το RIGHT JOIN φέρνει όλους τους καλλιτέχνες από τον πίνακα performers, ακόμη κι αν δεν έχουν συνδεδεμένα άλμπουμ στον πίνακα albums. Αν ένας καλλιτέχνης δεν έχει άλμπουμ (δηλαδή αν το performer_id του άλμπουμ δεν ταιριάζει με κανένα id στον πίνακα performers), τότε στα αποτελέσματα θα εμφανιστεί κανονικά το όνομα του καλλιτέχνη, αλλά οι στήλες Album και Genre θα είναι κενές (null).

4. Να βρεθούν όλα τα albums (τίτλος και είδος) που είναι καταχωρημένα στη βάση και ο αντίστοιχος καλλιτέχνης. Να εμφανίσετε επιπλέον τα albums που δεν αντιστοιχούν σε κάποιο καλλιτέχνη, καθώς και τους καλλιτέχνες που δεν έχουν κάποιο album.

```
SELECT albums.name AS Album, genre AS Genre, performers.name AS Artist
FROM albums
  LEFT JOIN performers ON albums.performer_id = performers.id
UNION
SELECT albums.name AS Album, genre AS Genre, performers.name AS Artist
FROM albums
  RIGHT JOIN performers ON albums.performer_id = performers.id
```

Το query αποτελείται από δύο μέρη:

- *Πρώτο Μέρος: Χρησιμοποιεί LEFT JOIN για να πάρει όλα τα άλμπουμ από τον πίνακα albums, ανεξάρτητα από το αν υπάρχει καλλιτέχνης που τα έχει δημιουργήσει. Αν κάποιο άλμπουμ δεν έχει συνδεδεμένο καλλιτέχνη, η τιμή της στήλης Artist θα είναι NULL.*
- *Δεύτερο Μέρος: Χρησιμοποιεί RIGHT JOIN για να πάρει όλους τους καλλιτέχνες από τον πίνακα performers, ακόμη κι αν δεν έχουν άλμπουμ. Αν κάποιος καλλιτέχνης δεν έχει συνδεδεμένο άλμπουμ, οι στήλες Album και Genre θα είναι NULL.*

Η εντολή UNION συνδυάζει τα αποτελέσματα των δύο τμημάτων του query, αφαιρώντας τα διπλά αποτελέσματα. Αυτό σημαίνει ότι:

- *Θα έχουμε στη λίστα μας όλα τα άλμπουμ και όλους τους καλλιτέχνες, ακόμα και αν κάποιο άλμπουμ δεν έχει καλλιτέχνη ή κάποιος καλλιτέχνης δεν έχει άλμπουμ.*
- *Η UNION διασφαλίζει ότι κάθε συνδυασμός άλμπουμ και καλλιτέχνη εμφανίζεται μόνο μία φορά.*

Ο συνδυασμός του LEFT JOIN και του RIGHT JOIN με την εντολή UNION έχει ίδιο αποτέλεσμα με ένα FULL OUTER JOIN

5. Να βρεθούν όλα τα albums (τίτλος και είδος) για τα οποία υπάρχει κάποιος καλλιτέχνης. Να εμφανίσετε μόνο τα albums και όχι τους καλλιτέχνες.

```
SELECT name, genre
FROM albums
WHERE EXISTS (
  SELECT *
  FROM performers
  WHERE albums.performer_id = performers.id
);
```

Το WHERE EXISTS χρησιμοποιείται γιατί ο έλεγχος του γίνεται για την ύπαρξη και μόνο της σχέσης, όχι για την ανάκτηση συγκεκριμένων δεδομένων από τον συνδεδεμένο πίνακα. Το αποτέλεσμα της πράξης στην παρένθεση θα γυρίζει TRUE ή FALSE.

6. Να βρεθούν όλα τα albums (τίτλος και είδος) για τα οποία υπάρχει κάποιος καλλιτέχνης. Να εμφανίσετε μόνο τα albums και όχι τους καλλιτέχνες. Αυτή τη φορά δεν επιτρέπεται να χρησιμοποιήσετε το keyword EXISTS στο ερώτημά σας.

```
SELECT name, genre
FROM albums
WHERE performer_id IN (
  SELECT id
  FROM performers
);
```

Η εντολή WHERE ... IN συγκρίνει το performer_id του κάθε άλμπουμ με όλα τα id που επιστρέφονται από το υποερώτημα. Αν το performer_id βρίσκεται στη λίστα με τα IDs των καλλιτεχνών, το άλμπουμ αυτό συμπεριλαμβάνεται στο τελικό αποτέλεσμα.

$$\{t \mid t \in \text{albums} \ \wedge \ \exists s \in \text{performers}(t[\text{performer\_id}] = s[\text{id}])\}$$

7. Να βρεθούν όλες οι εταιρείες παραγωγής (label) των οποίων υπάρχει τουλάχιστον ένα album καταχωρημένο στην ΒΔ και να εμφανιστούν ταξινομημένες με αλφαβητική σειρά.

```
SELECT DISTINCT label FROM albums
ORDER BY label ASC
```

Το query αποτελείται από δύο βασικά στοιχεία:

- *DISTINCT: Η λέξη-κλειδί DISTINCT εξασφαλίζει ότι θα εμφανιστεί μόνο μία φορά κάθε μοναδική τιμή για το πεδίο label. Αν υπάρχουν άλμπουμ από την ίδια δισκογραφική εταιρεία, το DISTINCT θα αφαιρέσει τα διπλά και θα δείξει μόνο μία εγγραφή για κάθε εταιρεία.*
- *ORDER BY ASC: Η εντολή ORDER BY label ASC ταξινομεί τα αποτελέσματα σε αύξουσα σειρά (από το Α στο Ω) σύμφωνα με το πεδίο label. Το ASC (Ascending) υποδεικνύει την αύξουσα σειρά, αλλά θα μπορούσε να παραληφθεί, καθώς η SQL την θεωρεί προεπιλογή.*

8. Να βρεθεί ο αριθμός των albums, τα οποία ηχογραφήθηκαν μετά το 1990.

```
SELECT COUNT(id) FROM albums
WHERE year_recorded > 1990
```

Το query χρησιμοποιεί δύο κύριες εντολές:

- *COUNT(id): Η εντολή COUNT(id) μετράει τον αριθμό των γραμμών όπου το πεδίο id έχει τιμή (δηλαδή δεν είναι NULL). Δεδομένου ότι το id είναι μοναδικό για κάθε άλμπουμ στον πίνακα albums, το COUNT(id) μας δίνει τον συνολικό αριθμό των άλμπουμ που πληρούν τις προϋποθέσεις του WHERE φίλτρου.*
- *WHERE year_recorded > 1990: Το WHERE φίλτρο περιορίζει τα αποτελέσματα στα άλμπουμ που ηχογραφήθηκαν μετά το 1990. Έτσι, το COUNT θα εφαρμόζεται μόνο σε αυτά τα άλμπουμ.*

9. Να βρεθεί το πιο πρόσφατο από τα έτη ηχογράφησης των καταχωρημένων στην ΒΔ album.

```
SELECT MAX(year_recorded)
FROM albums
```

Η εντολή MAX(year_recorded) επιστρέφει τη μεγαλύτερη τιμή στο πεδίο year_recorded. Αυτό σημαίνει ότι το query θα μας δώσει το πιο πρόσφατο έτος στο οποίο ηχογραφήθηκε κάποιο άλμπουμ στον πίνακα albums.

10. Να βρεθεί το πιο πρόσφατο από τα έτη ηχογράφησης των καταχωρημένων στην ΒΔ album για group, αλλά και solo καλλιτέχνες.

```
SELECT MAX(year_recorded), is_group
FROM albums JOIN performers ON albums.performer_id = performers.id
GROUP BY is_group
```

Το query χρησιμοποιεί ορισμένες βασικές εντολές:

- *JOIN performers ON albums.performer_id = performers.id: Αυτή η εντολή ενώνει τον πίνακα albums με τον πίνακα performers μέσω της σχέσης performer_id, ώστε να αποκτήσουμε πρόσβαση στη στήλη is_group από τον πίνακα performers. Έτσι, μπορούμε να γνωρίζουμε αν κάθε καλλιτέχνης είναι συγκρότημα ή μεμονωμένος.*
- *MAX(year_recorded): Η συνάρτηση MAX(year_recorded) βρίσκει το πιο πρόσφατο έτος ηχογράφησης για κάθε κατηγορία καλλιτέχνη (μεμονωμένος ή συγκρότημα). Αυτό σημαίνει ότι το query θα υπολογίσει τη μεγαλύτερη τιμή της στήλης year_recorded για κάθε ομάδα που δημιουργείται από την επόμενη εντολή GROUP BY.*
- *GROUP BY is_group: Αυτή η εντολή ομαδοποιεί τα αποτελέσματα σύμφωνα με το αν ο καλλιτέχνης είναι συγκρότημα ή όχι (is_group). Το GROUP BY διασφαλίζει ότι η συνάρτηση MAX θα εφαρμόζεται ξεχωριστά σε κάθε κατηγορία (is_group = TRUE και is_group = FALSE), παρέχοντας μας το πιο πρόσφατο έτος ηχογράφησης για group αλλά και για solo καλλιτέχνη.*

11. Να βρεθούν ο τίτλος, το είδος, το έτος ηχογράφησης και ο καλλιτέχνης (performer) του πιο πρόσφατου (σύμφωνα με το έτος ηχογράφησης) καταχωρημένου στην ΒΔ album.

```
SELECT albums.name, genre, performers.name, year_recorded
FROM albums JOIN performers ON albums.performer_id = performers.id
WHERE year_recorded IN
  (SELECT MAX(year_recorded) FROM albums)
```

JOIN performers ON albums.performer_id = performers.id: Αυτή η εντολή συνδέει τον πίνακα albums με τον πίνακα performers μέσω του performer_id. Έτσι, μπορούμε να ανακτήσουμε τις πληροφορίες του καλλιτέχνη που αντιστοιχούν σε κάθε άλμπουμ.

WHERE year_recorded IN (...): Η συνθήκη WHERE ... IN φιλτράρει τα άλμπουμ ώστε να περιλαμβάνονται μόνο εκείνα που έχουν ηχογραφηθεί το πιο πρόσφατο έτος.

Υποερώτημα SELECT MAX(year_recorded) FROM albums: Το υποερώτημα επιστρέφει το μεγαλύτερο έτος ηχογράφησης (MAX(year_recorded)) από τον πίνακα albums. Αυτή η τιμή χρησιμοποιείται ως φίλτρο στην κύρια ερώτηση.

12. Να βρεθούν τα ονόματα και η παλαιότητα των albums της βάσης και να ταξινομηθούν από το πιο σύγχρονο στο πιο παλιό. Η παλαιότητα να εκφραστεί σε χρόνια που πέρασαν μέχρι το 2024.

```
SELECT name, 2024 - year_recorded AS years_ago
FROM albums
ORDER BY years_ago
```

Το query χρησιμοποιεί τις ακόλουθες εντολές:

- *2024 - year_recorded AS years_ago: Αυτή η έκφραση υπολογίζει την ηλικία του κάθε άλμπουμ σε σχέση με το έτος 2024, αφαιρώντας την τιμή του πεδίου year_recorded από το 2024. Η τιμή αυτή εμφανίζεται ως years_ago, που μας δείχνει πόσα χρόνια έχουν περάσει από την ηχογράφησή του.*
- *ORDER BY years_ago: Η εντολή ORDER BY years_ago ταξινομεί τα αποτελέσματα κατά την ηλικία των άλμπουμ, ξεκινώντας από τα πιο πρόσφατα (λιγότερα χρόνια) και καταλήγοντας στα παλαιότερα.*

Η γενικευμένη προβολή (ή generalized projection) είναι ένας όρος που περιγράφει μια προβολή (δηλαδή, επιλογή συγκεκριμένων στηλών) που μπορεί να περιλαμβάνει όχι μόνο τα πεδία ενός πίνακα, αλλά και:

- *Συναρτήσεις και Υπολογισμούς – όπως μαθηματικές πράξεις (π.χ., πρόσθεση, αφαίρεση) ή συναρτήσεις όπως SUM, AVG, MAX, MIN.*
- *Συνθήκες και Φίλτρα – μέσω του WHERE ή και άλλων περιορισμών.*
- *Ομαδοποιήσεις – που εφαρμόζονται με τη χρήση του GROUP BY ώστε να μπορούμε να εκτελούμε υπολογισμούς σε ομάδες δεδομένων.*
- *Συνδυασμούς από Πίνακες – μέσω των JOIN, ώστε να μπορούμε να φέρουμε δεδομένα από πολλούς πίνακες σε μία προβολή.*

Σε μια γενικευμένη προβολή, λοιπόν, δεν επιλέγουμε απλώς συγκεκριμένα πεδία, αλλά μπορούμε να εκτελούμε και πιο πολύπλοκες πράξεις που εμπλουτίζουν το σύνολο δεδομένων που εμφανίζεται.