**Εθνικό Μετσόβιο Πολυτεχνείο**

Σχολή Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών

# Βάσεις Δεδομένων

Αναφορά Εξαμηνιαίας Εργασίας

2024-2025

## Ομάδα 77

Γιαννουσιάδης Ανδρόνικος – 03120241

Σουφλέρη Αλεξάνδρα – 03121637

Ημερομηνία παράδοσης: Τετάρτη, 14 Μαΐου 2025

https://github.com/AndronikosG/Pulse_Music_Festival.git

## Περιεχόμενα


Α : Σχεδιασμός και υλοποίηση της Βάσης Δεδομένων ..... 3

Α.1: Εισαγωγή ..... 3

Α.2: Διαγράμματα ..... 3

Α.2.1: ER Διάγραμμα ..... 3

Α.2.2: Σχεσιακό Διάγραμμα ..... 4

Α.3: Υλοποίηση της Βάσης ..... 4

Α.3.1: Πίνακες και περιορισμοί ..... 4

Α.3.1: Ευρετήρια (Indexes) ..... 6

Α.4: Εισαγωγή δεδομένων στη Βάση ..... 8

Β : Ερωτήματα (queries) ..... 9

Β.1:Query 4 ..... 9

Β.2: Query 6 ..... 11

# **A :** Σχεδιασμός και υλοποίηση της Βάσης Δεδομένων

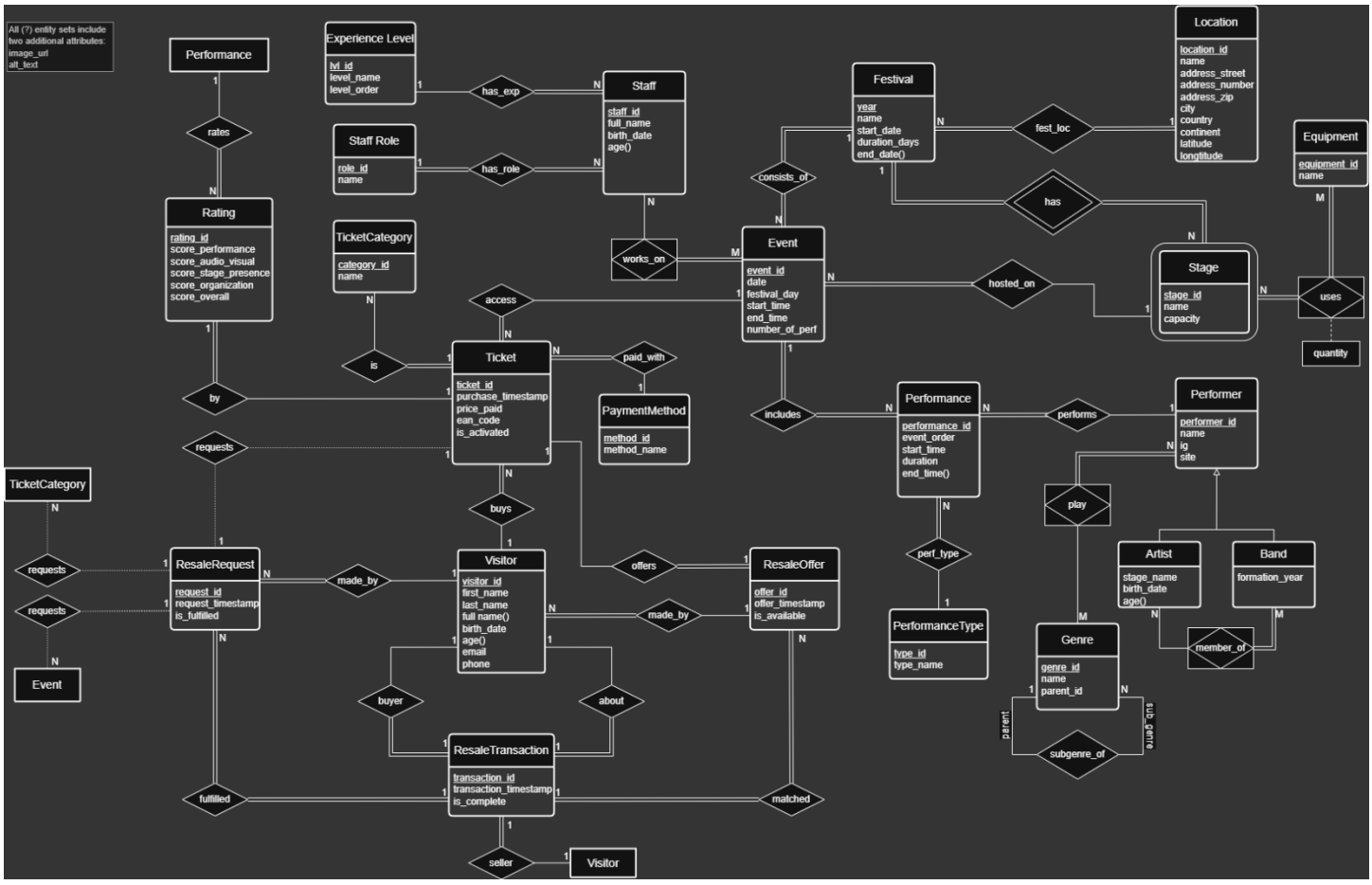
## A.1: Εισαγωγή

Η βάση δεδομένων υλοποιήθηκε με χρήση γλώσσας **MariaDB**(έκδοση: 10.4), μέσω **XAMPP**, χωρίς τη χρήση ORMs, enums, arrays, json και xml. Η δημιουργία του ER διαγράμματος έγινε μέσω της εφαρμογής diagrams.net (aka draw.io), ενώ το σχεσιακό διάγραμμα δημιουργήθηκε στην πλατφόρμα DBdiagram (με χρήση DBML). Τέλος, ως GUI χρησιμοποιήθηκε το phpMyAdmin.

Η παρούσα αναφορά περιγράφει αναλυτικά την διαδικασία σχεδιασμού και υλοποίησης, αναλύει τη λογική πίσω από τις επιλογές σχεδίασης και παρουσιάζει/απαντά τα δοσμένα από την εκφώνηση ερωτήματα (queries).

## A.2: Διαγράμματα

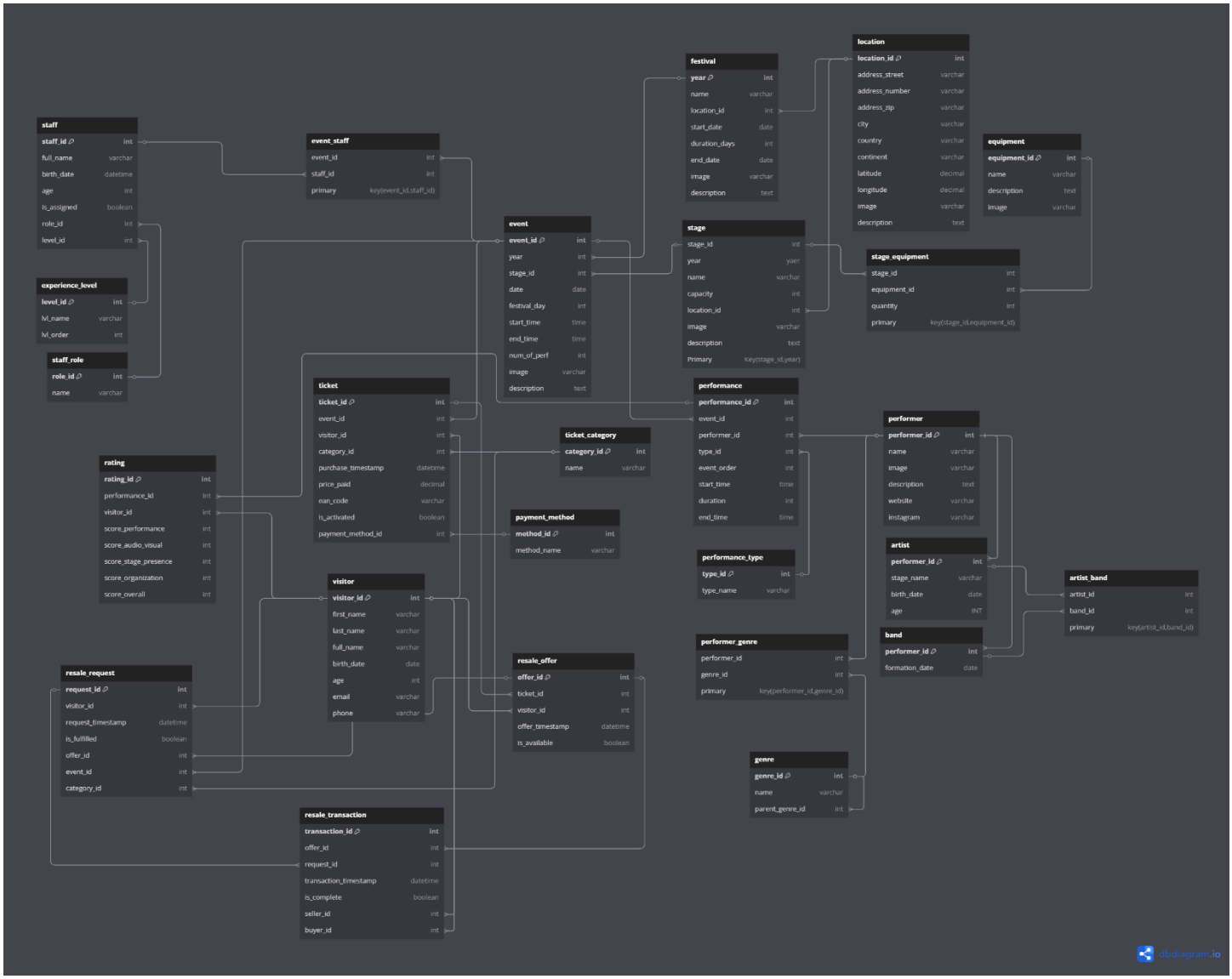
### A.2.1: ER Διάγραμμα




Όπως αναφέρεται και στο διάγραμμα (πάνω αριστερά) όλες οι οντότητες περιλαμβάνουν, πέραν των αναγραφόμενων, και τα attributes 'image' και 'description'.

Το διάγραμμα περιλαμβάνει 25+ οντότητες και ακολουθεί τις αρχές κανονικοποίησης και παρουσιάζει σαφώς τόσο τις οντότητες όσο και τις σχέσεις που τις συνδέουν, ενώ αποτυπώνονται ορθά περιπτώσεις αδύναμων οντοτήτων (*stage*), χρήσης εξειδίκευσης (*artist, band → performer*) και ενός recursive relationship (*genre*).

## A.2.2:Σχεσιακό Διάγραμμα




Το σχεσιακό διάγραμμα δημιουργήθηκε βάσει του ER διαγράμματος, διασφαλίζοντας την ακεραιότητα των δεδομένων, την αποφυγή πλεονασμών και τη δυνατότητα εκτέλεσης σύνθετων queries. Περιλαμβάνει αυτούσιες τις οντότητες του προηγούμενου διαγράμματος, μαζί με τις N-to-M σχέσεις (junction tables). Για λόγους "αποσυμφόρησης" του διαγράμματος, στα entity tables δεν είναι εμφανή τα constraints και indexes που περιλαμβάνονται στους ορισμούς τους. Επιπλέον, το πρόγραμμα που χρησιμοποιήθηκε δεν είχε δυνατότητα περαιτέρω συμβολισμού αδύναμων οντοτήτων, πέρα από το composite primary key που τις χαρακτηρίζει.

## A.3: Υλοποίηση της Βάσης

### A.3.1: Πίνακες και περιορισμοί

Στο αρχείο *install.sql* δημιουργούνται 27 πίνακες, η δομή και τα περιεχόμενα των οποίων φαίνονται αναλυτικά. Ορισμένες επιλογές, άξιες ιδιαίτερης μνείας, είναι οι εξής:

- Σχεδόν όλοι οι πίνακες έχουν ως πρωτεύον κλειδί ένα παραγόμενο αυτόματα id. Εξαιρέσεις αποτελούν ο πίνακας φεστιβάλ, που για πρωτεύον κλειδί έχει το έτος (εφόσον αναφέρεται κατηγορηματικά πως μπορεί να υπάρξει μόνο ένα φεστιβάλ τον χρόνο), αλλά και ο πίνακας stage. Ο τελευταίος έχει composite πρωτεύον κλειδί (stage_id, year). Με τον τρόπο αυτό επιτυγχάνεται ένας πιο ρεαλιστικός χαρακτηρισμός των σκηνών του εκάστοτε φεστιβάλ και αποτυπώνεται η σχέση εξάρτησης των εν λόγω σκηνών από την ύπαρξη του φεστιβάλ.
- Με δεδομένο, ότι απαγορεύεται η χρήση enums, όπου κρίθηκε αναγκαίο χρησιμοποιήθηκαν lookup tables (πχ staff_role, ticket_category κ.α.). Αυτό διευκολύνει την επεκτασιμότητα της βάσης (ενδεχόμενες προσθήκες νέων τιμών), βελτιώνει αναγνωσιμότητα του κώδικα και διευκολύνει τη διασύνδεση (JOIN) με άλλους πίνακες. Μοναδική εξαίρεση σε αυτό, αποτέλεσε η χρήση CHECK για τις τιμές της ηπείρου στον πίνακα location, μιας και δεν υπάρχει προοπτική προσθήκης νέας τιμής.
- Χρησιμοποιώντας την τεχνική της εξειδίκευσης, δημιουργήθηκε ένας ενιαίος πίνακας performer, ο οποίος διαχειρίζεται τις υπο-οντότητες (και τους αντίστοιχους πίνακες) artist και band. Η επιλογή αυτή, πέρα από λογική και ρεαλιστική, αποδεικνύεται ιδιαίτερα βοηθητική στον χειρισμό των queries.
- Το σύστημα μεταπώλησης με λογική FIFO επιτυγχάνεται με τη χρήση 3 πινάκων (βλέπε παρακάτω), ενός trigger που ενεργοποιεί το σύστημα μεταπώλησης σε περίπτωση εξάντλησης των εισιτηρίων για συγκεκριμένη παράσταση, ενός procedure που ελέγχει και διαχειρίζεται της προσφορές, τα αιτήματα και τα πιθανά ματσαρίσματα, εφαρμόζοντας την λογική FIFO, καθώς και ένα event το οποίο καλεί συνεχώς την εν λόγω procedure.
    - resale_offer: διαχείριση των εισιτηρίων προς μεταπώληση
    - resale_request: διαχείριση των αιτημάτων αγοράς των εν λόγω εισιτηρίων, είτε στοχεύοντας σε συγκεκριμένο εισιτήριο προς πώληση ή βάσει του συνδυασμού παράστασης και κατηγορίας εισιτηρίου.
    - resale_transaction: αποθήκευση των στοιχείων επιτυχημένων μεταπωλήσεων μέσω της PROCEDURE fifo_matching
- Τέλος, οι πληροφορίες για τις οποίες είναι εφικτό, υπολογίζονται αυτόματα, είτε εντός του πίνακα, όπου το επιτρέπει η SQL (πχ festival.end_date, performance.edn_time), είτε με χρήση κάποιου trigger (πχ staff.age, visitor.age)

Μετά τους ορισμούς των πινάκων, ακολουθούν triggers, μέσω των οποίων ικανοποιούνται περιορισμοί που ορίζονται από την εκφώνηση ή/και την λογική. Μεταξύ άλλων, αποτρέπεται η αλλαγή σε συγκεκριμένα στοιχεία εισιτηρίων, αποτρέπεται το overbooking VIP και συνολικών εισιτηρίων, απαγορεύεται η πώληση και μεταπώληση παλαιών ή ενεργοποιημένων εισιτηρίων, επιβάλλεται η διάρκεια των εμφανίσεων και των διαλειμμάτων αλλά και απαγορεύεται η συμμετοχή καλλιτεχνών σε φεστιβάλ για πάνω από τρία συναπτά έτη.

### A.3.1: Ευρετήρια (Indexes)

Εντός των ορισμών των πινάκων έχουν οριστεί και τα κατάλληλα ευρετήρια για να βελτιστοποιηθεί η απόδοση των ζητούμενων queries αλλά παράλληλα να μη επιβαρυνθεί συνολικά η απόδοση της βάσης. Καταρχάς, δεν υπάρχει ανάγκη δημιουργίας indexes για τιμές που είναι πρωτεύοντα κλειδιά ή ανήκουν σε περιορισμό UNIQUE(). Επιπλέον, και βάσει οδηγίας που έχει δοθεί και στο εργαστήριο, δημιουργήσαμε indexes για όλα τα foreign keys του κάθε πίνακα. Όλα τα υπόλοιπα εξυπηρετούν αποκλειστικά την διευκόλυνση των queries και παρουσιάζονται παρακάτω:

- Query 1: Έσοδα ανά μέθοδο πληρωμής και έτος
    - idx_ticket_year_payment ON ticket (event_id, payment_method_id)
      Σύνδεση εισιτηρίων με παραστάσεις και φιλτράρισμα ανά μέθοδο πληρωμής
    - idx_event_year ON event (event_id, year)
      Ανάκτηση του έτους μέσω event_id

- Query 2: Καλλιτέχνες ενός είδους & συμμετοχή σε φεστιβάλ
    - idx_genre_name ON genre (genre_name)
      Φιλτράρισμα βάσει μουσικού είδους
    - idx_perf_type_event_perrformer (type_id, event_id, performer_id)
      Έλεγχος εάν ένας performer έχει εμφανιστεί σε φεστιβάλ συγκεκριμένου έτους
    - idx_event_year ON event (event_id, year)
      Σύνδεση με έτος

- Query 3: Καλλιτέχνες ως Opening acts
    - idx_event_year ON event (event_id, year) (FK)
      Φιλτράρισμα opening εμφανίσεων
    - idx_event_year ON event (event_id, year)
      Σύνδεση με έτος

- Query 4: Μέσος όρος αξιολογήσεων για καλλιτέχνη
    - idx_rating_performance ON rating (performance_id) (FK)
      Σύνδεση αξιολόγησης με εμφάνιση
    - idx_performance_performer (performer_id, performance_id)
      Σύνδεση εμφάνισης με καλλιτέχνη

- Query 5: Νέοι καλλιτέχνες με πολλαπλές συμμετοχές
    - idx_artist_age ON artist (age)
      Επιλογή μόνο νέων (<30) καλλιτεχνών
    - idx_perf_performer ON performance (performer_id) (FK)
      Μέτρηση εμφανίσεων για κάθε καλλιτέχνη

- Query 6: Αριθμός παραστάσεων και μ.ο. αξιολογήσεων για επισκέπτη
    - idx_tick_vis_ev (visitor_id, event_id, is_activated)
      Εύρεση ενεργοποιημένων εισιτηρίων για τον επισκέπτη
    - idx_perf_event ON performance (event_id) (FK)
      Εύρεση των παραστάσεων που παρακολούθησε
    - idx_rating_ticket_perf ON rating (ticket_id, performance_id)
      Αντιστοίχιση αξιολόγησης με εισιτήριο και παράσταση

- Query 7: Μέσος όρος εμπειρίας τεχνικού προσωπικού
    - idx_staff_role ON staff (role_id) (FK)
      Επιλογή μόνο τεχνικού προσωπικού
    - idx_staff_exp ON staff (level_id) (FK)
      Εύρεση του επιπέδου εμπειρίας
    - idx_event_staff_event ON event_staff (event_id) (FK)
      Εύρεση προσωπικού ανά παράσταση

- Query 8: Υποστηρικτικό προσωπικό χωρίς προγραμματισμένη εργασία
    - idx_staff_role ON staff (role_id) (FK)
      Επιλογή μόνο υποστηρικτικού προσωπικού
    - idx_event_staff_staff ON event_staff (staff_id) (FK)
      Έλεγχος ύπαρξης προγραμματισμένης εργασίας
    - idx_event_date ON event (event_id, date)
      Έλεγχος για συγκεκριμένη ημερομηνία

- Query 9: Επισκέπτες με ίδιο αριθμό παραστάσεων >3
    - idx_tick_vis_ev (visitor_id, event_id, is_activated)
      Εύρεση ενεργοποιημένων εισιτηρίων για τον επισκέπτη
    - idx_perf_event ON performance (event_id, performance_id)
      Αντιστοίχιση εμφανίσεων με παραστάσεις

- Query 10: Top-3 ζεύγη μουσικών ειδών με πολλές εμφανίσεις
    - idx_perf_performer ON performance (performer_id) (FK)
      Μέτρηση εμφανίσεων για κάθε καλλιτέχνη

- Query 11: Καλλιτέχνες με τουλάχιστον 5 λιγότερες εμφανίσεις από αυτόν με τις περισσότερες
    - idx_perf_performer ON performance (performer_id) (FK)
      Μέτρηση εμφανίσεων για κάθε καλλιτέχνη

- Query 12: Απαιτούμενο προσωπικό ανά ημερομηνία φεστιβάλ
    - idx_event_day_stage (year, date, stage_id)
      Εύρεση σκηνών ανά ημέρα
    - idx_stage_capacity ON stage (stage_id, year, capacity)
      Ανάκτηση χωρητικότητας σκηνής και υπολογισμός προσωπικού

- Query 13: Καλλιτέχνες σε 3+ ηπείρους
    - idx_perf_performer_event ON performance (performer_id, event_id)
      Σύνδεση καλλιτέχνη με παράσταση
    - idx_event_year ON event (event_id, year)
      Σύνδεση παράστασης με φεστιβάλ
    - idx_festival_location ON festival (year, location_id)
      Σύνδεση φεστιβάλ με τοποθεσία
    - idx_location_continent ON location (location_id, continent)
      Ανάκτηση ηπείρου

- Query 14: Μουσικά είδη με ίδιο αριθμό εμφανίσεων σε διαδοχικά έτη
    - idx_perf_event_performer ON performance (performer_id, event_id)
      Σύνδεση καλλιτεχνών με παραστάσεις
    - idx_event_year ON event (event_id, year)
      Ομαδοποίηση ανά έτος

- Query 15: Top-5 επισκέπτες με υψηλότερη συνολική βαθμολόγηση σε καλλιτέχνη
    - idx_rating_ticket_perf ON rating (ticket_id, performance_id)
      Αντιστοίχιση αξιολόγησης με εμφάνιση και εισιτήριο
    - idx_ticket_visitor ON ticket (ticket_id, visitor_id)
      Αντιστοίχιση επισκέπτη με εισιτήριο
    - idx_perf_performer ON performance (performance_id, performer_id)
      Αντιστοίχιση καλλιτέχνη με εμφάνιση

## A.4: Εισαγωγή δεδομένων στη Βάση

Για την παραγωγή δεδομένων χρησιμοποιήθηκε κατά βάση το Mockaroo, ενώ σε συγκεκριμένες περιπτώσεις έγινε χρήση LLM (ChatGPT) για παραγωγή είτε πιο ρεαλιστικών και έγκυρων δεδομένων (πχ καλλιτέχνες) είτε πιο σύνθετων δεδομένων που έπρεπε να ικανοποιούν δεδομένες συνθήκες (πχ performer_genre). Να σημειωθεί πως οι μόνοι πίνακες που δεν απαιτούν manual εισαγωγή δεδομένων είναι οι event_staff και resale_transaction, των οποίον τα δεδομένα εισάγονται από τα procedures assign_staff_for_festival([year]) (απαιτεί manual call) και fifo_matching() (αυτόματα μέσω event) αντίστοιχα. Τέλος τα δεδομένα για το σύστημα μεταπώλησης βρίσκονται στο τέλος του *load.sql* σε μορφή σχολίου για την εξυπηρέτηση πιο στοχευμένων δοκιμών.

# **B :** Ερωτήματα (queries)

Για κάθε ερώτημα υπάρχει και ένα αντίστοιχο query (*Qxx.sql*) καθώς και ένα αρχείο κειμένου με ότι αυτό επέστρεψε (*Qxx_out.txt*), στον φάκελο sql\.

Για τα ερωτήματα 4 και 6, καλούμαστε να παρουσιάσουμε εναλλακτικά query plans και να δοκιμάσουμε διαφορετικές στρατηγικές join. Καταρχάς, βάσει της αδυναμίας μας να τρέξουμε συγκεκριμένες εντολές, σε συνδυασμό με την έρευνα στην οποία οδηγηθήκαμε, συμπεραίνουμε πως η έκδοση MariaDB που χρησιμοποιούμε δεν υποστηρίζει hash ή merge joins. Συνεπώς δε μπορούμε να ικανοποιήσουμε αυτό το κομμάτι της εκφώνησης.
Σχετικά με τα εναλλακτικά query plans δοκιμάσαμε τα εξής:

## B.1: Query 4

| Αρχικό query | Εναλλακτικό query plan |
|---|---|
| SELECT<br>pf.performer_id,<br>COALESCE(a.stage_name, p.name) AS performer_name,<br>ROUND(AVG(r.score_performance), 2) AS avg_score_performance,<br>ROUND(AVG(r.score_overall), 2) AS avg_score_overall<br>FROM rating r<br>JOIN performance pf ON r.performance_id = pf.performance_id<br>JOIN performer p ON pf.performer_id = p.performer_id<br>LEFT JOIN artist a ON p.performer_id = a.performer_id<br>LEFT JOIN band b ON p.performer_id = b.performer_id<br>WHERE pf.performer_id = 31 -- Replace with the desired artist ID<br>GROUP BY pf.performer_id, performer_name; | SELECT<br>pf.performer_id,<br>COALESCE(a.stage_name, p.name) AS performer_name,<br>ROUND(AVG(r.score_performance), 2) AS avg_score_performance,<br>ROUND(AVG(r.score_overall), 2) AS avg_score_overall<br>FROM rating r FORCE INDEX (idx_rating_perf)<br>JOIN performance pf ON r.performance_id = pf.performance_id<br>JOIN performer p ON pf.performer_id = p.performer_id<br>LEFT JOIN artist a ON p.performer_id = a.performer_id<br>LEFT JOIN band b ON p.performer_id = b.performer_id<br>WHERE pf.performer_id = 31<br>GROUP BY pf.performer_id, performer_name;<br>G\<br><br>(Force index idx_rating_perf) |

Παρατηρούμε πως ο optimizer επέλεξε το index performance_id για τον πίνακα rating, με access type ref και εκτιμώμενα r_rows = 25 και r_total_time_ms ≈ 0.0668 ms

Χρησιμοποιώντας την εντολή ANALYZE FORMAT=JSON παίρνουμε το παρακάτω αποτέλεσμα:

```
ANALYZE: {
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 0.0988,
    "table": {
      "table_name": "p",
      "access_type": "const",
      "possible_keys": ["PRIMARY"],
      "key": "PRIMARY",
      "key_length": "4",
      "used_key_parts": ["performer_id"],
      "ref": ["const"],
      "r_loops": 0,
      "rows": 1,
      "r_rows": null,
      "filtered": 100,
      "r_filtered": null
    },
    "table": {
      "table_name": "a",
      "access_type": "const",
      "possible_keys": ["PRIMARY", "idx_artist_perfor
      "key": "PRIMARY",
      "key_length": "4",
      "used_key_parts": ["performer_id"],
      "ref": ["const"],
      "r_loops": 0,
      "rows": 1,
      "r_rows": null,
      "filtered": 100,
      "r_filtered": null
    },
    "table": {
      "table_name": "pf",
      "access_type": "ref",
      "possible_keys": [
        "PRIMARY",
        "idx_perf_performer",
        "idx_perf_performer_event",
        "idx_performance_performer"
      ],
      "key": "idx_perf_performer",
      "key_length": "4",
      "used_key_parts": ["performer_id"],
      "ref": ["const"],
      "r_loops": 1,
      "rows": 3,
      "r_rows": 3,
      "r_total_time_ms": 0.0039,
      "filtered": 100,
      "r_filtered": 100,
      "using_index": true
    },
    "table": {
      "table_name": "r",
      "access_type": "ref",
      "possible_keys": ["performance_id", "idx_rating
      "key": "performance_id",
      "key_length": "4",
      "used_key_parts": ["performance_id"],
      "ref": ["pulse_uni.pf.performance_id"],
      "r_loops": 3,
      "rows": 8,
      "r_rows": 25,
      "r_total_time_ms": 0.0668,
      "filtered": 100,
      "r_filtered": 100
    }
  }
}
1 row in set (0.001 sec)
```

```
query_block": {
 "select_id": 1,
 "r_loops": 1,
 "r_total_time_ms": 0.1092,
 "table": {
   "table_name": "p",
   "access_type": "const",
   "possible_keys": ["PRIMARY"],
   "key": "PRIMARY",
   "key_length": "4",
   "used_key_parts": ["performer_id"],
   "ref": ["const"],
   "r_loops": 0,
   "rows": 1,
   "r_rows": null,
   "filtered": 100,
   "r_filtered": null
 },
 "table": {
   "table_name": "a",
   "access_type": "const",
   "possible_keys": ["PRIMARY", "idx_artist_performer"],
   "key": "PRIMARY",
   "key_length": "4",
   "used_key_parts": ["performer_id"],
   "ref": ["const"],
   "r_loops": 0,
   "rows": 1,
   "r_rows": null,
   "filtered": 100,
   "r_filtered": null
 },
 "table": {
   "table_name": "pf",
   "access_type": "ref",
   "possible_keys": [
     "PRIMARY",
     "idx_perf_performer",
     "idx_perf_performer_event",
     "idx_performance_performer"
   ],
   "key": "idx_perf_performer",
   "key_length": "4",
   "used_key_parts": ["performer_id"],
   "ref": ["const"],
   "r_loops": 1,
   "rows": 3,
   "r_rows": 3,
   "r_total_time_ms": 0.0033,
   "filtered": 100,
   "r_filtered": 100,
   "using_index": true
 },
 "table": {
   "table_name": "r",
   "access_type": "ref",
   "possible_keys": ["idx_rating_perf"],
   "key": "idx_rating_perf",
   "key_length": "4",
   "used_key_parts": ["performance_id"],
   "ref": ["pulse_uni.pf.performance_id"],
   "r_loops": 3,
   "rows": 8,
   "r_rows": 25,
   "r_total_time_ms": 0.081,
   "filtered": 100,
   "r_filtered": 100
 }
```

Βλέπουμε ότι η προσπάθεια επιβολής του συγκεκριμένου index όχι απλώς δε βελτιώνει την απόδοση αλλά αυξάνει ελαφρώς τον χρόνο εκτέλεσης (~0.081 ms). Συνεπώς, το αρχικό query plan, στο οποίο ο optimizer επιλέγει από μόνος του την performance_id index στο rating, είναι ήδη βελτιστοποιημένο.

## B.2: Query 6

Παρομοίως και για το query 6

| Αρχικό query | Εναλλακτικό query plan |
|---|---|
| SELECT<br>pf.performer_id,<br>COALESCE(a.stage_name, p.name) AS performer,<br>ROUND((<br>r.score_performance +<br>r.score_audio_visual +<br>r.score_stage_presence +<br>r.score_organization +<br>r.score_overall<br>) / 5, 2) AS average_score<br>FROM ticket t<br>JOIN event e ON t.event_id = e.event_id<br>JOIN performance pf ON e.event_id = pf.event_id<br>JOIN performer p ON pf.performer_id = p.performer_id<br>LEFT JOIN artist a ON p.performer_id = a.performer_id<br>LEFT JOIN rating r ON pf.performance_id = r.performance_id<br>AND t.ticket_id = r.ticket_id<br>WHERE t.visitor_id = 1 -- Replace with the desired visitor ID<br>AND t.is_activated = TRUE<br>ORDER BY pf.performance_id; | SELECT<br>pf.performer_id,<br>COALESCE(a.stage_name, p.name) AS performer,<br>ROUND((<br>r.score_performance +<br>r.score_audio_visual +<br>r.score_stage_presence +<br>r.score_organization +<br>r.score_overall<br>) / 5, 2) AS average_score<br>FROM ticket t<br>JOIN event e ON t.event_id = e.event_id<br>JOIN performance pf ON e.event_id = pf.event_id<br>JOIN performer p ON pf.performer_id = p.performer_id<br>LEFT JOIN artist a ON p.performer_id = a.performer_id<br>LEFT JOIN rating r FORCE INDEX (idx_rating_ticket_perf)<br>ON pf.performance_id = r.performance_id<br>AND t.ticket_id = r.ticket_id<br>WHERE t.visitor_id = 1 -- Replace with the desired visitor ID<br>AND t.is_activated = TRUE<br>ORDER BY pf.performance_id;<br><br>(Force index idx_rating_ticket_perf) |

Παρατηρούμε πως ο optimizer επέλεξε το index performance_id για τον πίνακα rating, με access type eq_ref και εκτιμώμενα rows = 1 και r_total_time_ms ≈ 0.0087 ms

Χρησιμοποιώντας την εντολή ANALYZE FORMAT=JSON παίρνουμε το παρακάτω αποτέλεσμα:

```
{
"query_block": {
  "select_id": 1,
  "r_loops": 1,
  "r_total_time_ms": 1.2903,
  "filesort": {
    "sort_key": "pf.performance_id",
    "r_loops": 1,
    "r_total_time_ms": 0.0051,
    "r_used_priority_queue": false,
    "r_output_rows": 2,
    "r_buffer_size": "300",
    "temporary_table": {
      "table": {
        "table_name": "t",
        "access_type": "ref",
        "possible_keys": [
          "visitor_id",
          "idx_ticket_event",
          "idx_ticket_visitor",
          "idx_tick_vis_ev",
          "idx_ticket_event_pm_price"
        ],
        "key": "idx_tick_vis_ev",
        "key_length": "4",
        "used_key_parts": ["visitor_id"],
        "ref": ["const"],
        "r_loops": 1,
        "rows": 3,
        "r_rows": 3,
        "r_total_time_ms": 0.0113,
        "filtered": 100,
        "r_filtered": 33.333,
        "attached_condition": "t.is_activated = 1",
        "using_index": true
      },
      "table": {
        "table_name": "e",
        "access_type": "eq_ref",
        "possible_keys": ["PRIMARY",
         "idx_event_year",
         "idx_event_date"],
        "key": "PRIMARY",
        "key_length": "4",
        "used_key_parts": ["event_id"],
        "ref": ["pulse_uni.t.event_id"],
        "r_loops": 1,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 1.1782,
        "filtered": 100,
        "r_filtered": 100,
        "using_index": true
      },
      "table": {
        "table_name": "pf",
        "access_type": "ref",
        "possible_keys": [
          "event_id",
          "event_id_2",
          "idx_perf_event",
          "idx_perf_performer",
          "idx_perf_performer_event",
          "idx_performance_performer"
        ],
        "key": "event_id_2",
        "key_length": "4",
        "used_key_parts": ["event_id"],
        "ref": ["pulse_uni.t.event_id"],
        "r_loops": 1,
        "rows": 1,
        "r_rows": 2,
        "r_total_time_ms": 0.0323,
        "filtered": 100,
        "r_filtered": 100,
        "using_index": true
      },
```

```
{
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 0.0766,
    "filesort": {
      "sort_key": "pf.performance_id",
      "r_loops": 1,
      "r_total_time_ms": 0.0055,
      "r_used_priority_queue": false,
      "r_output_rows": 2,
      "r_buffer_size": "300",
      "temporary_table": {
        "table": {
          "table_name": "t",
          "access_type": "ref",
          "possible_keys": [
            "visitor_id",
            "idx_ticket_event",
            "idx_ticket_visitor",
            "idx_tick_vis_ev",
            "idx_ticket_event_pm_price"
          ],
          "key": "idx_tick_vis_ev",
          "key_length": "4",
          "used_key_parts": ["visitor_id"],
          "ref": ["const"],
          "r_loops": 1,
          "rows": 3,
          "r_rows": 3,
          "r_total_time_ms": 0.0081,
          "filtered": 100,
          "r_filtered": 33.333,
          "attached_condition": "t.is_activated = 1",
          "using_index": true
        },
        "table": {
          "table_name": "e",
          "access_type": "eq_ref",
          "possible_keys": ["PRIMARY",
          "idx_event_year",
           "idx_event_date"],
          "key": "PRIMARY",
          "key_length": "4",
          "used_key_parts": ["event_id"],
          "ref": ["pulse_uni.t.event_id"],
          "r_loops": 1,
          "rows": 1,
          "r_rows": 1,
          "r_total_time_ms": 0.0034,
          "filtered": 100,
          "r_filtered": 100,
          "using_index": true
        },
        "table": {
          "table_name": "pf",
          "access_type": "ref",
          "possible_keys": [
            "event_id",
            "event_id_2",
            "idx_perf_event",
            "idx_perf_performer",
            "idx_perf_performer_event",
            "idx_performance_performer"
          ],
          "key": "event_id_2",
          "key_length": "4",
          "used_key_parts": ["event_id"],
          "ref": ["pulse_uni.t.event_id"],
          "r_loops": 1,
          "rows": 1,
          "r_rows": 2,
          "r_total_time_ms": 0.0031,
          "filtered": 100,
          "r_filtered": 100,
          "using_index": true
        },
```

```
      "table": {
        "table_name": "p",
        "access_type": "eq_ref",
        "possible_keys": ["PRIMARY"],
        "key": "PRIMARY",
        "key_length": "4",
        "used_key_parts": ["performer_id"],
        "ref": ["pulse_uni.pf.performer_id"],
        "r_loops": 2,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 0.004,
        "filtered": 100,
        "r_filtered": 100
      },
      "table": {
        "table_name": "a",
        "access_type": "eq_ref",
        "possible_keys": ["PRIMARY", "idx_artist_performer"],
        "key": "PRIMARY",
        "key_length": "4",
        "used_key_parts": ["performer_id"],
        "ref": ["pulse_uni.pf.performer_id"],
        "r_loops": 2,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 0.0026,
        "filtered": 100,
        "r_filtered": 100
      },
      "table": {
        "table_name": "r",
        "access_type": "eq_ref",
        "possible_keys": [
          "performance_id",
          "idx_rating_ticket_perf",
          "idx_rating_perf",
          "idx_rating_ticket"
        ],
        "key": "performance_id",
        "key_length": "8",
        "used_key_parts": ["performance_id"
        , "ticket_id"],
        "ref": ["pulse_uni.pf.performance_id"
        , "pulse_uni.t.ticket_id"],
        "r_loops": 2,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 0.0087,
        "filtered": 100,
        "r_filtered": 100
      }
    }
  }
}
```



```
      "table": {
        "table_name": "p",
        "access_type": "eq_ref",
        "possible_keys": ["PRIMARY"],
        "key": "PRIMARY",
        "key_length": "4",
        "used_key_parts": ["performer_id"],
        "ref": ["pulse_uni.pf.performer_id"],
        "r_loops": 2,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 0.006,
        "filtered": 100,
        "r_filtered": 100
      },
      "table": {
        "table_name": "a",
        "access_type": "eq_ref",
        "possible_keys": ["PRIMARY",
         "idx_artist_performer"],
        "key": "PRIMARY",
        "key_length": "4",
        "used_key_parts": ["performer_id"],
        "ref": ["pulse_uni.pf.performer_id"],
        "r_loops": 2,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 0.0027,
        "filtered": 100,
        "r_filtered": 100
      },
      "table": {
        "table_name": "r",
        "access_type": "ref",
        "possible_keys": ["idx_rating_ticket_perf"],
        "key": "idx_rating_ticket_perf",
        "key_length": "8",
        "used_key_parts": ["ticket_id",
         "performance_id"],
        "ref": ["pulse_uni.t.ticket_id",
         "pulse_uni.pf.performance_id"],
        "r_loops": 2,
        "rows": 1,
        "r_rows": 1,
        "r_total_time_ms": 0.01,
        "filtered": 100,
        "r_filtered": 100
      }
    }
  }
}
```


Βλέπουμε πως και σε αυτή την περίπτωση ο optimizer επέλεξε το καταλληλότερο index (performance_id), ενώ η επιβολή του idx_rating_ticket_perf δεν οδήγησε σε καλύτερη απόδοση (~0.01 ms) επομένως το αρχικό query plan θεωρείται πιο αποτελεσματικό.