

## Extragere Interogări SQL din Codul Sursă

Şova Ioan-Rares

Grupa 332AA

## Aplicație pentru evidența spectacolelor de Stand-Up

### A. Interogări Simple (cu JOIN)

Cerință proiect: Minim 6 interogări simple care presupun JOIN, cu parametri variabili.

#### 1. Lista principală de spectacole (JOIN pe 3 tabele)

Afișarea tuturor spectacolelor disponibile, înlocuind ID-urile cu numele real al locației și al organizatorului, cu posibilitate de căutare dinamică (parametru variabil) și sortare.

```
208     );
209     String column = safeColumns.getOrDefault(sortBy.toUpperCase(), defaultValue: "s.Data_Spectacol");
210     String direction = "DESC".equalsIgnoreCase(sortDir) ? "DESC" : "ASC";
211
212     String sql = "SELECT s.ID_Spectacol, s.Titlu, s.Data_Spectacol, s.Ora, s.Pret_Bilet, "
213             + "l.Nume_Locatie, o.Nume_Organizator, s.ID_Locatie, s.ID_Organizator "
214             + "FROM Spectacol s LEFT JOIN Locatie l ON s.ID_Locatie = l.ID_Locatie LEFT JOIN Organizator o ON s.ID_Organizator = o.ID_Organizator";
215
216     boolean hasSearch = !search.isEmpty();
217     if (hasSearch) sql += " WHERE s.Titlu LIKE ? OR l.Nume_Locatie LIKE ? OR o.Nume_Organizator LIKE ?";
218
219     sql += " ORDER BY " + column + " " + direction;
```

#### 2. Istorul biletelor vândute (JOIN pe 3 tabele)

Afișarea biletelor cumpărate, legând tabelele Bilet, Spectacol și Spectator pentru a arăta detalii complete. Include filtrare parametrică după ID-ul spectatorului conectat. (parametru variabil)

```
374     String column = safeColumns.getOrDefault(sortBy.toUpperCase(), defaultValue: "b.Data_Cumparare");
375     String direction = "DESC".equalsIgnoreCase(sortDir) ? "DESC" : "ASC";
376
377     String sql = "SELECT b.ID_Bilet, s.Titlu, sp.Nume_Spectator, b.Data_Cumparare, b.Cod_Bilet "
378             + "FROM Bilet b LEFT JOIN Spectacol s ON b.ID_Spectacol = s.ID_Spectacol LEFT JOIN Spectator sp ON b.ID_Spectator = sp.ID_Spectator";
379     if (!"admin".equalsIgnoreCase(ses.role)) sql += " WHERE b.ID_Spectator = ? ";
380     sql += " ORDER BY " + column + " " + direction;
```

#### 3. Widget: Ultimele bilete vândute (Admin Dashboard)

Afișarea ultimelor 3 tranzacții înregistrate în sistem pentru panoul de administrare.

```
401
402     ignoreCase(ses.role)) {
403         t = "SELECT TOP 3 b.Cod_Bilet, s.Titlu, sp.Nume_Spectator FROM Bilet b "
404             + "JOIN Spectacol s ON b.ID_Spectacol = s.ID_Spectacol JOIN Spectator sp ON b.ID_Spectator = sp.ID_Spectator ORDER BY b.Data_Cumparare DESC";
405     st = conn.createStatement(); ResultSet rsRec = st.executeQuery(sqlRecent);
406     ring, Object>> recentTickets = new ArrayList<>();
407     c.next() {
```

#### 4. Widget: Următoarele evenimente (Pagina Locații)

Identificarea următoarelor 5 spectacole programate în viitor, folosind funcția de dată curentă (GETDATE).

```

544
545
546     sqlUp = "SELECT TOP 5 l.Nume_Locatie, s.Titlu, s.Data_Spectacol FROM Locatie l JOIN Spectacol s ON l.ID_Locatie=s.ID_Locatie WHERE s
547     statement st = conn.createStatement(); ResultSet rs = st.executeQuery(sqlUp) {
548         t<Map<String, Object>> up = new ArrayList<>();
549         while(rs.next()) {
550
551             l JOIN Spectacol s ON l.ID_Locatie=s.ID_Locatie WHERE s.Data_Spectacol >= CAST(GETDATE() AS DATE) ORDER BY s.Data_Spectacol ASC";
552         }
553
554

```

## 5. Widget: Colaborări recente (Pagina Organizator)

Afișarea ultimelor 5 evenimente organizate, ordonate descrescător după dată.

```

724     }
725 }
726 String sqlRec = "SELECT TOP 5 o.Nume_Organizator, s.Titlu FROM Organizator o JOIN Spectacol s ON o.ID_Organizator=s.ID_Organizator
727     try (Statement st = conn.createStatement(); ResultSet rs = st.executeQuery(sqlRec)) {
728         List<Map<String, Object>> rec = new ArrayList<>();
729
730         while(rs.next()) {
731             o.Nume_Organizator, s.Titlu FROM Organizator o JOIN Spectacol s ON o.ID_Organizator=s.ID_Organizator ORDER BY s.Data_Spectacol DESC";
732             statement st = conn.createStatement(); ResultSet rs = st.executeQuery(sqlRec) {
733                 m=new ArrayList<>();
734                 Object> m=new HashMap<>(); m.put("org", rs.getString( columnIndex: 1)); m.put("show", rs.getString( columnIndex: 2)); rec.add(m); }
735             m.put("recentShows", rec);

```

## 6. Listă de control rapidă (Raport)

O listă simplificată a ultimelor spectacole adăugate și a locațiilor aferente.

```

511
512     = "SELECT TOP 5 s.Titlu, l.Nume_Locatie FROM Spectacol s JOIN Locatie l ON s.ID_Locatie=l.ID_Locatie ORDER BY s.ID_Spectacol DESC";
513     checklist = new ArrayList<>();
514     t st = conn.createStatement(); ResultSet rs = st.executeQuery(sql6){
515         next(); checklist.add(rs.getString( columnIndex: 1) + " (" + rs.getString( columnIndex: 2) + ")");

```

## B. Interogări Complexe

Cerință proiect: Minim 4 interogări complexe (subcereri, funcții agregat, operatori), cu parametri variabili.

### 1. Raport Vânzări Totale

Calculul numărului de bilete vândute și a sumei totale încasate pentru fiecare spectacol, filtrat într-un interval de timp specificat de utilizator (parametri variabili startDate, endDate).

```

437
438     // COMPLEXA 1
439     String sql1 = "SELECT T.Titlu, T.Nr, T.Total FROM " +
440         "(SELECT s.Titlu, COUNT(b.ID_Bilet) AS Nr, SUM(s.Pret_Bilet) AS Total " +
441         "FROM Spectacol s LEFT JOIN Bilet b ON s.ID_Spectacol = b.ID_Spectacol " +
442         "WHERE s.Data_Spectacol BETWEEN ? AND ? " +
443         "GROUP BY s.ID_Spectacol, s.Titlu) AS T " +
444         "WHERE T.Nr > 0 ORDER BY T.Total DESC";
445

```

### 2. Top Organizatori (Subcerere în WHERE)

Clasamentul organizatorilor în funcție de veniturile generate, luând în calcul doar organizatorii care au spectacole active.

```
457  
458 // COMPLEXA 2  
459 String sql2 = "SELECT o.Nume_Organizator, SUM(s.Pret_Bilet) AS Total " +  
    "FROM Organizator o LEFT JOIN Spectacol s ON o.ID_Organizator=s.ID_Organizator " +  
    "LEFT JOIN Bilet b ON s.ID_Spectacol=b.ID_Spectacol " +  
    "WHERE o.ID_Organizator IN (SELECT ID_Organizator FROM Spectacol) " +  
    "GROUP BY o.Nume_Organizator" +  
    "HAVING SUM(s.Pret_Bilet) > 0 " +  
    "ORDER BY Total DESC";  
460  
461  
462  
463  
464  
465  
466
```

### 3. Cel mai fidel spectator (Subcerere Scalară + TOP 1)

Identificarea spectatorului cu cele mai multe bilete achiziționate. Utilizează o subcerere scalară în lista de selecție și o subcerere cu TOP 1 pentru filtrare.

```
472  
473 // COMPLEXA 3  
474 String sql3 = "SELECT sp.Nume_Spectator " +  
    "(SELECT COUNT(*) FROM Bilet b WHERE b.ID_Spectator = sp.ID_Spectator) AS Cnt " +  
    "FROM Spectator sp " +  
    "WHERE sp.ID_Spectator = (SELECT TOP 1 ID_Spectator FROM Bilet GROUP BY ID_Spectator ORDER BY COUNT(*) DESC)";  
475  
476  
477  
478
```

### 4. Gradul de Ocupare (Subcereri Corelate)

Calculul statistic pentru fiecare locație: numărul de show-uri găzduite și numărul total de bilete vândute, pentru a determina eficiența sălii.

```
485 // COMPLEXA 4  
486 String sql4 = "SELECT l.Nume_Locatie, l.Capacitate, " +  
    "(SELECT COUNT(*) FROM Spectacol s WHERE s.ID_Locatie = l.ID_Locatie) AS NrShow, " +  
    "(SELECT COUNT(*) FROM Bilet b JOIN Spectacol s ON b.ID_Spectacol=s.ID_Spectacol WHERE s.ID_Locatie=l.ID_Locatie) AS Vandute " +  
    "FROM Locatie l WHERE l.Capacitate > 0";  
487  
488
```