FIȘĂ DE LUCRU · CLASA A XII-A · FIȘA 2 · ACOPERĂ LECȚIILE XII.4–XII.5

# SQL: CREATE, INSERT, UPDATE, DELETE, SELECT

Durată: 100 min  Nivel: mate-info

**CE VEI ȘTI DUPĂ LECȚIE**

- Prezici rezultatul unui SELECT cu WHERE, LIKE, BETWEEN, IN, IS NULL, DISTINCT, ORDER BY.
- Găsești greșelile din CREATE, INSERT, UPDATE și DELETE, inclusiv pe cele pe care SGBD-ul nu le semnalează.
- Scrii comenzi care creează, completează și modifică tabele fără să strici datele.

## Ce trebuie să știi (o pagină)

| Comandă / operator | Exemplu pe baza ȘCOALA | Efect |
|---|---|---|
| `CREATE TABLE` | `CREATE TABLE t (id INTEGER PRIMARY KEY, nume VARCHAR(30) NOT NULL UNIQUE, id_elev INTEGER REFERENCES elevi(id_elev));` | tabel nou; întâi tabelele „părinte” |
| cheie compusă | `PRIMARY KEY (id_elev, id_club)` | perechea nu se repetă |
| `ALTER TABLE` | `ALTER TABLE elevi ADD email VARCHAR(80);` | coloană nouă (NULL peste tot) |
| `DROP TABLE` | `DROP TABLE note;` | dispare tot, fără „Undo” |
| `INSERT` | `INSERT INTO clase (id_clasa, nume_clasa) VALUES (6, 'X A');` | coloanele nenumite primesc NULL |
| `UPDATE` | `UPDATE note SET nota = nota + 1 WHERE id_nota = 7;` | modifică rândurile din WHERE |
| `DELETE` | `DELETE FROM note WHERE nota < 5;` | șterge rândurile din WHERE |
| ordinea în `SELECT` | `SELECT … FROM … WHERE … ORDER BY …` | fixă |
| `DISTINCT` / `ORDER BY` | `SELECT DISTINCT profil FROM clase ORDER BY profil DESC;` | fără dubluri / sortare |
| `LIKE` | `'M%'` începe cu M; `'_o%'` a doua literă o; `'____'` exact 4 caractere | `%` = orice șir, `_` = un caracter |
| `BETWEEN a AND b` | `nota BETWEEN 5 AND 7` | **inclusiv** capetele; `a ≤ b` |

| Comandă / operator | Exemplu pe baza ȘCOALA | Efect |
|---|---|---|
| `IN (…)` | `id_disc IN (1, 3)` | la fel ca `id_disc = 1 OR id_disc = 3` |
| `IS NULL` | `data_nasterii IS NULL` | `= NULL` nu e niciodată adevărat |
| prioritate | `AND` înaintea lui `OR` | pune paranteze |
| alias, calcul | `SELECT nota * 10 AS procent FROM note;` | coloană calculată |
| concatenare | SQLite: `nume \|\| ' ' \|\| prenume`; MySQL: `CONCAT(nume, ' ', prenume)` | text lipit |

Datele sunt scrise ca text `'AAAA-LL-ZZ'`, deci se pot compara (`<`, `BETWEEN`) și filtra cu `LIKE '2026-09%'`. Înainte de orice exercițiu care **modifică** date, rulează din nou `scoala.sql` (începe cu `DROP TABLE`, deci reface baza de la zero).

> **ATENȚIE — GREȘELI FRECVENTE**
>
> `UPDATE` și `DELETE` **fără WHERE** lucrează pe tot tabelul. Rulează întâi `SELECT * FROM … WHERE …` cu același `WHERE` și uită-te ce rânduri vor fi afectate.

## Nivelul 1 · Prezice

★ citește și prezice

**P1.** Ce afișează fiecare interogare (în ordine)?

```sql
SELECT nume FROM elevi WHERE id_clasa = 3 ORDER BY nume;
SELECT prenume FROM elevi WHERE id_clasa = 3 ORDER BY nume DESC;
```

**P2. Tiparele LIKE.** Ce nume (sau prenume) se potrivesc? Completează tabelul.

| condiția | rezultate |
|---|---|
| `nume LIKE 'M%'` | |
| `nume LIKE '%escu'` | |
| `nume LIKE '_o%'` | |
| `nume LIKE '%a'` | |
| `prenume LIKE 'a%'` (cu **a** mic!) | |
| `prenume LIKE '____'` | |

**P3.** Câte rânduri dă fiecare interogare? Câte dau **împreună**? De ce nu 10?

```sql
SELECT nume FROM elevi WHERE data_nasterii < '2010-01-01';
SELECT nume FROM elevi WHERE NOT (data_nasterii < '2010-01-01');
```

**P4.** Ce `id_nota` afișează prima interogare? Dar a doua?

```sql
SELECT id_nota FROM note WHERE data_nota BETWEEN '2026-10-01' AND '2026-10-05';
SELECT id_nota FROM note WHERE data_nota BETWEEN '2026-10-05' AND '2026-10-01';
```

**P5.** Ce afișează `SELECT DISTINCT id_clasa FROM elevi ORDER BY id_clasa DESC;`? Una dintre clase lipsește. Care și de ce?

**P6.** Ce `id_nota` dă fiecare? Care corespunde cerinței „notele de cel puțin 9 la Fizică (4) sau Română (3)”?

```sql
SELECT id_nota FROM note WHERE id_disc = 4 OR id_disc = 3 AND nota >= 9;
SELECT id_nota FROM note WHERE (id_disc = 4 OR id_disc = 3) AND nota >= 9;
```

**P7.** Pe baza proaspătă se rulează **de două ori** aceeași comandă. Câte rânduri modifică prima rulare? Dar a doua? Ce note au la final `id_nota` 7, 15 și 17?

```sql
UPDATE note SET nota = nota + 1 WHERE nota < 5;
```

**P8.** Completează rezultatul (atenție la împărțire!):

```sql
SELECT id_nota, nota, nota * 10 AS procent, nota / 2 AS jumatate, nota / 2.0 AS jumatate2
FROM note WHERE id_nota IN (7, 15);
```

## Nivelul 2 · Găsește și repară

★★ găsește și repară

**P9.** „Elevii fără dată de naștere” dă **0 rânduri**, deși Barbu Tudor nu are dată. De ce? Repară.

```sql
SELECT nume FROM elevi WHERE data_nasterii = NULL;
```

**P10.** Trei greșeli. SQLite o semnalează pe prima la `CREATE`, pe a doua abia la primul `INSERT`, iar pe a treia **niciodată**. Găsește-le pe toate.

```sql
CREATE TABLE cluburi (
    id_club   INTEGER PRIMARY KEY,
    nume      VARCHAR(30) NOT NULL UNIQUE
    profesor  VARCHAR(40)
);
CREATE TABLE membri (
    id_elev  INTEGER REFERENCES elevi(id_elev),
    id_club  INTEGER REFERENCES club(id_club),
    PRIMARY KEY (id_elev)
);
```

**P11. Rânduri amestecate.** Construiește interogarea „notele de cel puțin 9 la Informatică (`id_disc = 2`), cele mai noi primele”. Un rând e în plus. Apoi prezice rezultatul.

```
A)  ORDER BY data_nota DESC;
B)  SELECT id_elev, nota
C)  WHERE id_disc = 2
D)  FROM note
E)  AND nota >= 9
F)  WHERE nota >= 9
```

**P12.** Dirigintele clasei X B devine „Stan”. Doi elevi au scris variantele de mai jos. Câte rânduri modifică fiecare? Ce s-a întâmplat? Scrie varianta corectă.

```sql
UPDATE clase SET diriginte = 'Stan';
UPDATE clase SET diriginte = 'Stan' WHERE nume_clasa = 'XB';
```

**P13.** `DELETE FROM elevi WHERE id_elev = 7;` dă eroarea „FOREIGN KEY constraint failed”. De ce? Scrie comenzile care o șterg pe Moraru Elena corect. Ce s-ar fi întâmplat fără verificarea cheilor străine?

**P14. Ce greșeală a făcut elevul?** Pentru fiecare comandă: e refuzată (și de ce) sau acceptată? Una e acceptată, dar strică interogările ulterioare. Care și cum?

```sql
a) INSERT INTO elevi VALUES (11, 'Pop', 'Ion', 3);
b) INSERT INTO elevi VALUES (11, Pop, 'Ion', NULL, 3);
c) INSERT INTO elevi VALUES (11, 'Pop', 'Ion', '14.03.2010', 3);
```

## Nivelul 3 · Scrie interogarea

★★ scrie cod

Lucrează pe baza ȘCOALA (reîncărcată). Interogarea e gata doar când dă **exact** rezultatul din test.

**P15. La mulți ani!** Elevii născuți în octombrie, noiembrie sau decembrie (în orice an), în ordinea zilei de naștere din an. Indiciu: `LIKE` cu `_` și `substr(data_nasterii, 6)`.

| nume | prenume | data_nasterii |
|---|---|---|
| Ionescu | Matei | 2009-11-02 |
| Moraru | Elena | 2008-12-19 |

**P16. Alarma de toamnă.** Notele din septembrie 2026 mai mici decât 7, la Matematică sau Română, de la cea mai mică la cea mai mare (coloane: `id_nota, id_elev, nota`).

| id_nota | id_elev | nota |
|---|---|---|
| 15 | 7 | 3 |
| 7 | 3 | 4 |
| 9 | 4 | 6 |

**P17. Cluburile școlii.** Creează tabelele CLUBURI (`id_club`, `nume` unic și obligatoriu, `profesor`) și INSCRIERI (`id_elev`, `id_club`, `data_inscrierii`), cu cheile potrivite. Adaugă cluburile Robotică (1, Ilie) și Șah (2, Pop). Testează (cu `PRAGMA foreign_keys = ON`):

| comanda | trebuie să fie |
|---|---|
| `INSERT INTO inscrieri VALUES (1, 1, '2026-10-01');` | acceptată |
| `INSERT INTO inscrieri VALUES (1, 2, '2026-10-02');` | acceptată |
| `INSERT INTO inscrieri VALUES (1, 1, '2026-10-20');` | refuzată |
| `INSERT INTO inscrieri VALUES (99, 1, '2026-10-20');` | refuzată |

| comanda | trebuie să fie |
|---|---|
| `INSERT INTO cluburi VALUES (3, 'Șah', 'Ene');` | refuzată |

**P18. Adrese de e-mail.** Adaugă în ELEVI coloana `email` și completeaz-o pentru toți elevii cu **o singură** comandă, după regula `prenume.nume@scoala.ro`, cu litere mici (funcția `lower`). Test: elevul 1 → `ana.popescu@scoala.ro`, elevul 10 → `tudor.barbu@scoala.ro`.

**P19. Corecția profesorului de fizică.** Toate notele la Fizică (`id_disc = 4`) din octombrie 2026 cresc cu 1 punct, dar nicio notă nu poate trece de 10 (și nici nu trebuie să dea eroare). Test: după comandă, `SELECT id_nota, nota FROM note WHERE id_disc = 4;` dă `10 6`, `12 10`, `19 10`, iar comanda raportează **2** rânduri modificate.

**P20. Plecări.** Moraru Elena și Barbu Tudor s-au transferat. Șterge-i din bază, cu tot ce ține de ei, fără să dezactivezi cheile străine. Află id-urile cu un `SELECT` după nume. Test: `SELECT id_elev FROM elevi;` dă 8 rânduri, iar `SELECT * FROM note WHERE id_elev = 7;` nu dă niciunul.

## Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

**P21. Problema inversă.** Găsește **trei** condiții `WHERE` diferite (cu cel mult două condiții legate prin `AND` fiecare, fără `id_elev` și fără `OR`) care dau exact: Marin Ioana, Munteanu Maria, Matei Sara.

**P22. Rulat de două ori.** Ce se întâmplă dacă rulezi fiecare comandă de **două** ori la rând? Care dă același rezultat ca o singură rulare?

```
a) INSERT INTO discipline VALUES (5, 'Biologie');
b) INSERT INTO jurnal (mesaj) VALUES ('backup');   -- jurnal(id INTEGER PRIMARY KEY, mesaj TEXT)
c) UPDATE clase SET diriginte = 'Stan' WHERE id_clasa = 3;
d) UPDATE note SET nota = nota - 1 WHERE id_elev = 5;
e) DELETE FROM note WHERE nota < 5;
```

**P23. Interschimbare.** La Barbu Tudor, numele și prenumele au fost trecute invers. Ce face comanda de mai jos în SQLite? Ai fi zis că merge? (Află ce face aceeași comandă în MySQL.)

```
UPDATE elevi SET nume = prenume, prenume = nume WHERE id_elev = 10;
```

**P24. Ce tip are telefonul?** Un elev adaugă coloana `telefon INTEGER` și salvează `'0722123456'`. Ce se afișează la `SELECT telefon …`? Și dacă tipul ar fi `VARCHAR(10)`, dar valoarea ar fi scrisă **fără** ghilimele? Ce regulă scoți de aici?

**P25. Copia de siguranță.** Înainte de modificări riscante, un elev face `CREATE TABLE note_copie AS SELECT * FROM note;`. Datele sunt copiate. Ce **nu** se copiază? Arată cu un `INSERT` care ar fi refuzat în NOTE, dar e acceptat în NOTE_COPIE.

---