„De la Realitate la Date,

de la Date, la Baze de Date

De la Baze de Date, la aplicatii Web sau Desktop,

De la interogari (SQL), la SENS,

la înțelegerea fenomenului studiat din Realitate

și la *decizii bazate de informație din prelucrare* de

Date din Realitate”

Prelegerea Nr.1 Laboratorul Nr. 1

De la Realitate, *“poveste / sarcină”* , la Date

De la Realitate, *“poveste / sarcină”* , la Date
Studiu de caz/ Exemplu

- Gestionarea caracteristicilor unui PC - procesor, memorie, disc și temperatură - asigură performanțe optime, longevitatea sistemului și previne defecțiunile. Sarcinile cheie includ monitorizarea utilizării resurselor cu ajutorul Managerului de activități pentru a identifica blocajele, defragmentarea sau optimizarea unităților pentru a îmbunătăți performanța discului, curățarea fișierelor de sistem și gestionarea programelor de pornire pentru a elibera memorie, precum și asigurarea unui flux de aer adecvat și curățarea prafului pentru a menține temperaturi optime ale componentelor.

De la Date, la Baze de Date

Studiu de caz/ Exemplu - continuare

Pentru a menține funcționarea stabilă și durabilă a PC-ului, a preveni și diminua riscurile si menține performanța lui, ***este necesar să dezvoltăm un Sistem Infomatic,*** care să utilizeza Tehnologiile informaționale moderne cum ar fi:

- ***S***istemele de ***G***estiune a ***B***azelor de ***D***ate (***SGBD***) ***structurate / cu schemă fixă,*** (***SQLite, Mysql, MS SQL Server***), nestructurate (MongoDB) si a
- limbajelor de programare **(Python, Streamlit, PHP, Tkinter, Flask)**.

De la Date, la Baze de Date
Studiu de caz/ Exemplu - continuare

***SCOPUL** lucrării de laborator este de a crea / dezvolta un sistem informatic privind gestionarea caracteristicilor PC-ului pentru monitorizarea si administrarea lor, si a asigura performanță, longevitate, fiabilitate, stabilitate și securitate funcționării PC-ului.*

SCOPUL formulat poate fi atins prin următoarele *OBIECTIVE*:

IN CONTINUARE voi mentiona că structura **iesirilor/output-ului,** formularea si formatul lui depind în principiu, de comanda **SQL - SELECT...,** ***deoarece prin ea vom formula interogările la BD*** pentru a le obține!!!

Obiectivul 1 - pentru Performanță:

Menținerea componentelor reci, prin **monitorizarea temperaturii** care urmează să fie poziționată în intervale predefinite, să permită funcționarea PC-ului la eficiență maximă.

Structura unui SELECT în SQL pentru acest obiectiv

De la Date, la Baze de Date
Studiu de caz/ Exemplu - continuare

Obiectivul 2 - pentru Longevitate:

Căldura excesivă sau utilizarea constantă a resurselor pot duce la defectarea prematură a componentelor. Gestionarea eficientă prelungește durata de viață a hardware-ului, reducând necesitatea unor înlocuiri costisitoare.

Structura unui SELECT în SQL pentru acest obiectiv

Obiectivul 3– pentru Fiabilitate și stabilitate:

Gestionarea resurselor previne blocarea sau instabilitatea sistemului. ***Supraîncălzirea*** poate provoca funcționarea defectuoasă a componentelor, ducând la erori și pierderi de date.

Structura unui SELECT în SQL pentru acest obiectiv

Obiectivul 4– pentru Securitate:

Prin menținerea software-ului și a driverelor actualizate și ***gestionarea spațiului pe disc***, reducem și vulnerabilitățile de securitate.

Structura unui SELECT în SQL pentru acest obiectiv

Pentru atingerea obiectivelor, este necesar să fie rezolvate următoarele sarcini de bază cum ar fi:

Monitorizarea și gestionarea utilizării CPU/memoriei

Sarcină: Folosiți ***Produsul Informatic creat*** pentru a monitoriza utilizarea CPU, a memoriei si discului.

Structura unor SELECT în SQL pentru această sarcină

Soluții / Acțiuni:

Identificați procesele care consumă resurse mari și închideți aplicațiile inutile.

Gestionați programele de pornire pentru a le împiedica să se lanseze odată cu PC-ul, eliberând resurse.

Menținerea sănătății discului

Sarcină: Folosiți utilitare precum ***Gestionare disc*** sau ***Curățare disc***.

Soluții / Acțiuni:

Curățare disc: Eliminați fișierele temporare și alte date inutile pentru a elibera spațiu de stocare.

Defragmentare/Optimizare: Pentru hard disk-urile tradiționale (HDD-uri), defragmentarea acestora poate îmbunătăți performanța; pentru unitățile SSD (Solid State Drive), serviciile de optimizare efectuează sarcini similare.

Pentru atingerea obiectivelor, este necesar să fie rezolvate următoarele sarcini de bază cum ar fi:

Controlul și monitorizarea temperaturii

Sarcină: Folosiți ***Produsul Informatic creat*** pentru monitorizarea temperaturilor componentelor utilizând instrumente terțe sau BIOS-ul.

Structura unor SELECT în SQL pentru această sarcină

Soluții / Acțiuni:

Curățați PC-ul*: Îndepărtați regulat praful de pe ventilatoare și radiatoare pentru a asigura un flux de aer adecvat.*

Verificați ventilatoarele*: Asigurați-vă că toate ventilatoarele carcasei și ventilatoarele CPU/GPU funcționează corect.*

Îmbunătățiți fluxul de aer: *Asigurați-vă că PC-ul are o ventilație neobstrucționată pentru a preveni acumularea de căldură.*

Actualizarea și optimizarea sistemului de operare și driverele

Sarcină: Mențineți Windows și driverele de dispozitiv (de exemplu, pentru CPU, GPU) actualizate.

Soluții / Acțiuni:

Instalați actualizări ale sistemului de operare pentru a obține îmbunătățiri de performanță și remedieri de erori.

Actualizați driverele pentru a vă asigura că acestea comunică eficient cu hardware-ul și sistemul de operare.

KPI-uri (Key Performance Indicators)

Vom utiliza **KPI-urile (Key Performance Indicators)** **ca punct de plecare pentru formularea sarcinii, scopului și obiectivelo**r este o metodă **extrem de puternică** mai ales în contextul educației tehnice. Iată de ce:

✅ De ce are această abordare drept la existență (și chiar este excelentă)

1. KPI – ul se ancorează în realitate

KPI-urile sunt indicatori concreți, măsurabili, folosiți în industrie. Când studentul pornește de la un KPI, el nu mai lucrează cu „date abstracte”, ci cu **semnale reale** despre performanță, stabilitate, securitate etc.

2. KPI – ul activează gândirea analitică

Formularea scopului și obiectivelor pe baza KPI-urilor îi mobilizează pe proiectanți să gândească sistemic:

- Ce înseamnă acest indicator?
- Ce îl influențează?
- Cum pot interveni?

3. KPI – ul oferă autonomie, abordare individuală...

Fiecare student având propria sarcină = devine **proprietar al unei probleme REALE**. El, nu mai rezolvă „exerciții”, ci **proiecte personale cu SENS.**

4. KPI – ul leagă perfect sarcina cu SQL și programarea

KPI-ul devine o întrebare. Interogarea SQL devine răspunsul. Aplicația devine acțiunea.

Sugestii pentru rafinarea abordării KPI

1. KPI – ul este utilizat ca o întrebare creativă

Transformăm fiecare KPI într-o întrebare existențială:

1. *Temperatura CPU > 90°C* → „Respiră sistemul meu prea greu?”
2. *Memorie > 80%* → „Este sistemul meu copleșit de gânduri?”
3. *Disk usage > 90%* → „Mai are loc să-și amintească?”

Astfel, ***inginerul/proiectantul/studentul*** nu doar interoghează, ci și **reflectă**.

2. KPI → Scop → Obiectiv → Interogare → Recomandare

Un șablon modular pentru fiecare ***inginer/proiectant/student (Exemplu)***:

Etapă	Exemplu
KPI	CPU usage > 85%
Scop	Menținerea performanței sistemului
Obiectiv	Identificarea proceselor care suprasolicită CPU
Interogare SQL	`SELECT model, usage FROM CPU WHERE usage > 85;`
Recomandare	Închide aplicațiile inutile, optimizează pornirea

Ghid pentru formularea sarcinii pornind de la KPI
Studiu de caz/ Exemplu – continuare

Scopul Ghidului

Să ajute fiecare *inginer/proiectant/student* să transforme un **KPI real** într-o **sarcină clară**, cu un ***SCOP*** **personalizat**, o serie de **OBIECTIVE** **măsurabile**, și apoi să le obțină cu o **interogări SQL** ***dintr-o aplicație concretă*** urmată de o ***acțiune digitală***.

Structura Ghidului

1. Alege un KPI relevant

Exemple:

- CPU usage > 85%
- Temperatură GPU > 90°C
- Memorie RAM > 80%
- Spațiu pe disc < 10%

2. Formulează sarcina

„Monitorizarea și gestionarea [componenta] pentru a menține [KPI-ul] în limite optime.”

3. Definește *SCOPUL*

„Asigurarea performanței / stabilității / securității sistemului prin controlul [KPI].”

Ghid pentru formularea sarcinii pornind de la KPI
Studiu de caz/ Exemplu – continuare

4. Stabilește OBIECTIVELE

1. ***Identificarea valorilor critice***
2. ***Prevenirea suprasolicitării***
3. ***Optimizarea funcționării***

5. Scrie interogările SQL prin care pot fi obținute obiectivele, adică obținute răspunsurile la intrebarile "Ce doriim?". "De ce avem nevoie?"/

SQL

```
SELECT model, usage
FROM [tabela]
JOIN PC ON [tabela].id_pc = PC.id_pc
WHERE usage > [valoare critică];
```

6. Propune acțiuni digitale /decizii bazate pe SENS, obtinut prin obiective!!!/

1. ***Închide aplicații inutile***
2. ***Curăță sistemul***
3. ***Optimizează pornirea***

7. Reflectează

„Ce-mi spune acest KPI despre sănătatea sistemului meu cercetat?"

10 Șabloane de KPI-uri

Studiu de caz/ Exemplu – continuare

- 1 KPI: CPU usage > 85%
- 2 KPI: Memorie RAM > 80%
- 3 KPI: Spațiu pe disc < 10%
- 4 KPI: Temperatură CPU > 90°C
- 5 KPI: Temperatură GPU > 85°C
- 6 KPI: Număr de procese active > 100
- 7 KPI: CPU usage constant > 70% timp de 10 minute
- 8 KPI: SSD usage > 95%
- 9 KPI: Temperatură medie > 75°C
- 10 KPI: KPI: Memorie liberă < 20%

Plan modular introductiv –*KPI: De la semnal la soluție* , Studiu de caz/ Exemplu – continuare

Ce este un KPI?

Un **Key Performance Indicator** este un semnal. Un semnal că sistemul tău funcționează bine. Sau că are nevoie de ajutor.

Ce facem la acest pas? (EXEMPLU)

1. *Alegem un KPI real (ex: CPU usage > 85%)*
2. *Îl transformăm într-o sarcină clară*
3. *Formulăm un scop și obiective*
4. *Scriem interogări SQL care răspund la întrebare*
5. *Construim un dashboard interactiv care oferă soluții*

Fiecare student este un constructor de sens

Nu învățați doar SQL. Învățați să întrebați lumea digitală ce are nevoie. Și să răspundeți cu claritate, cu grijă, cu sens.

Structura gândirii

Etapă	Exemplu
KPI	CPU usage > 85%
Sarcină	Monitorizarea CPU
Scop	Menținerea performanței
Obiectiv	Identificarea suprasolicitării
Interogare SQL	SELECT model, usage FROM CPU WHERE usage > 85;
Recomandare	Închide aplicații inutile

De la Interogare la Înțelegere: Construim punți între REALITATE și DATE, , Studiu de caz/ Exemplu – continuare

Introducere motivațională. Proiectul, "poveste/text".

Fiecare calculator are componente vitale — procesor, memorie, disc, temperatură — care trebuie monitorizate și gestionate pentru ca sistemul să funcționeze bine, să dureze mult și să nu se blocheze. **Sarcina noastră este să creăm un sistem informatic care să ajute la monitorizarea acestor componente și să ofere soluții pentru optimizarea lor.**

Scopul Proiectului

Vrem să dezvoltăm o aplicație, care să ne ajute să înțelegem starea unui PC și să luăm decizii inteligente/înțelepte pentru:

- ***Performanță*** (să funcționeze rapid și eficient),
- ***Longevitate*** (să nu se strice repede),
- ***Fiabilitate*** (să nu se blocheze sau piardă date),
- ***Securitate*** (să nu fie vulnerabil la atacuri sau erori).

Reformulare : „De la nevoi reale la soluții digitale” , Studiu de caz/ Exemplu – continuare

Sarcina reală (Ce trebuie să facem?)

Fiecare calculator are componente vitale — procesor, memorie, disc, temperatură — care trebuie monitorizate și gestionate pentru ca sistemul să funcționeze bine, să dureze mult și să nu se blocheze. **Sarcina noastră este să creăm un sistem informatic care să ajute la monitorizarea acestor componente și să ofere soluții pentru optimizarea lor.**

Scopul proiectului (Ce dorim să obținem?)

Vrem să dezvoltăm o aplicație care să ne ajute să înțelegem starea unui PC și să luăm decizii inteligente pentru:

1. ***Performanță*** (să funcționeze rapid și eficient),
2. ***Longevitate*** (să nu se strice repede),
3. ***Fiabilitate*** (să nu se blocheze sau piardă date),
4. ***Securitate*** (să nu fie vulnerabil la atacuri sau erori).

Obiectivele (Cum atingem scopul?) , Studiu de caz/ Exemplu – continuare

Performanță

Ce dorim: Să menținem componentele reci și eficiente. **Cum facem:** Interogăm temperatura componentelor și verificăm dacă sunt în intervalul optim.

sql
```
SELECT model, component, value
FROM Temperatura
JOIN PC ON Temperatura.id_pc = PC.id_pc
WHERE value BETWEEN 30 AND 70;
```

Longevitate

Ce dorim: Să evităm uzura excesivă. **Cum facem:** Verificăm dacă procesorul este suprasolicitat și luăm măsuri.

sql
```
SELECT model, usage
FROM CPU
JOIN PC ON CPU.id_pc = PC.id_pc
WHERE usage < 50;
```

Obiectivele (Cum atingem scopul?) ,
Studiu de caz/ Exemplu – continuare

Fiabilitate și stabilitate

Ce dorim: Să prevenim blocajele și erorile. **Cum facem:** Monitorizăm memoria și identificăm suprasarcinile.

```
sql
SELECT model, usage
FROM Memorie
JOIN PC ON Memorie.id_pc = PC.id_pc
WHERE usage > 80;
```

Securitate

Ce dorim: Să evităm vulnerabilitățile. **Cum facem:** Verificăm spațiul pe disc și actualizările de sistem.

```
sql
SELECT model, usage
FROM Disk
JOIN PC ON Disk.id_pc = PC.id_pc
WHERE usage > 90;
```

Cum se leagă toate aceste idei si abordări?

Frontend (ce vede utilizatorul): Interfață creată cu Python, Streamlit, Flask, Tkinter sau PHP — unde utilizatorul poate vedea datele și lua decizii.

Backend (ce se întâmplă în spate): Baza de date SQLite, unde sunt stocate informațiile despre PC și unde se fac interogările SQL.

Liantul: Comenzile SELECT, INSERT, UPDATE, DELETE sunt puntea dintre ce vrem să aflăm și cum obținem răspunsul.

Mesaj final pentru ingineri/proiectanți/studenți

„Nu învățați SQL ca pe o limbă străină. Învățați-l ca pe un mod de a pune întrebări despre lumea digitală. Fiecare interogare e o întrebare ce urmează să găsească răspunsul in lumea reală. Fiecare tabelă e o parte dintr-un sistem viu ce reflectă o entitate/obiect sau mai multe, ce interaxționează între ele. Fiecare aplicație pe care o dezvoltăm este o soluție la o nevoie reală.

Plan motivațional pentru lucr. de laborator Nr.1

Laboratorul ca punte între gândire și acțiune

Ce trebuie făcut? Să înțelegem cum funcționează un PC și cum îl putem proteja.

Ce dorim? Să construim o aplicație care ne ajută să monitorizăm, să prevenim și să optimizăm.

Cum facem? Prin interogări SQL clare și aplicații frontend care transformă datele în decizii.

Fiecare interogare SQL este o întrebare despre REALITATE

SELECT = Ce vreau să aflu?

WHERE = Ce condiții pun ca să fie relevant?

JOIN = Cum leg realitățile între ele?

Fiecare inginer/proiectant/student este un constructor de punți

Între date și sens. Între sistem și decizie. Între ce trebuie făcut și cum se face.

Modulul 1: Cu ce începem? Cu Modelarea, Studiu de caz/ Exemplu – continuare

Sarcina reală Un calculator respiră prin procesor, memorie, disc și temperatură. Când ele sunt în echilibru, sistemul trăiește, când sunt ignorate, sistemul se sufocă si *sboieste*....

Scopul lucrării

Crearea unui sistem informatic pentru **monitorizarea și administrarea caracteristicilor PC-ului**: procesor, memorie, disc și temperatură — pentru a asigura performanță, longevitate, fiabilitate, stabilitate și securitate.

SCOPUL nostru *Să construim o aplicație care ascultă aceste respirații, Care le înțelege, Care le transformă în decizii,*

Studiu de caz/ Exemplu – continuare

Obiectivele noastre

1. ***Să păstrăm temperatura în limitele vieții.***
2. ***Să protejăm resursele de uzură.***
3. ***Să prevenim blocajele și erorile.***
4. ***Să păstrăm sistemul curat și sigur.***

Instrumentele noastre

SQL: limbajul întrebărilor.

Python, Streamlit, Flask, Tkinter, PHP: limbajele răspunsurilor.

Abordarea boastră

1. ***Scriem un SELECT.***
2. ***Punem un WHERE.***
3. ***Legăm cu JOIN.***
4. ***Validăm cu ochii și cu gândul.***

„Nu învățați doar să interogați o bază de date. Învățați să întrebați lumea digitală ce are nevoie. Și să răspundeți cu claritate, cu grijă, cu sens.”

Modulul 1: Modelarea,
Studiu de caz/ Exemplu – continuare

Obiective și interogări SQL

1 Performanță

SQL

```
SELECT model, component, value
FROM PC
JOIN Temperatura ON PC.id_pc =
Temperatura.id_pc
WHERE value BETWEEN 30 AND
70;
```

2 Longevitate

SQL

```
SELECT model, usage
FROM CPU
JOIN PC ON CPU.id_pc = PC.id_pc
WHERE usage < 50;
```

Obiective și interogări SQL

3 Fiabilitate și stabilitate

SQL

```
SELECT model, usage
FROM Memorie
JOIN PC ON Memorie.id_pc = PC.id_pc
WHERE usage > 80;
```

4 Securitate

SQL

```
SELECT model, usage
FROM Disk
JOIN PC ON Disk.id_pc = PC.id_pc
WHERE usage > 90;
```

LAB. Nr.1: *De la Interogare la Înțelegere,* Studiu de caz/ Exemplu – continuare

Ce trebuie făcut? Monitorizarea componentelor PC: CPU, memorie, disc, temperatură.

Ce dorim? Să construim o aplicație care oferă performanță, longevitate, fiabilitate și securitate.

Cum facem?

Backend: SQLite + interogări SQL (SELECT, JOIN, WHERE, CRUD)

Frontend: Python, Streamlit, Flask, Tkinter, PHP

Modulul 1: Modelarea Sarcini practice. SQL, Studiu de caz/ Exemplu – continuare

✅ **Monitorizarea CPU/Memorie/Disc**

```
SELECT model, usage FROM CPU JOIN PC ON CPU.id_pc = PC.id_pc;
```

```
SELECT model, usage FROM Memorie JOIN PC ON Memorie.id_pc = PC.id_pc;
```

```
SELECT model, usage FROM Disk JOIN PC ON Disk.id_pc = PC.id_pc;
```

✅ **Controlul temperaturii**

```
SELECT model, component, value FROM Temperatura JOIN PC ON Temperatura.id_pc = PC.id_pc;
```

Modulul 1: Modelarea Reflecție finală SQL
Studiu de caz/ Exemplu – continuare

Reflecție finală

Fiecare interogare SQL este o întrebare despre **REALITATE**. Fiecare tabelă este o reflecție a unei componente din **REALITATE**. Fiecare limbaj este o fereastră spre înțelegere. Fiecare inginer / student este un constructor de punți între **DATE** și **SENS**.

Fiind ingineri/proiectanți/studenți Dvs sunteți modelatori și constructori ai propriei vieți digitale. Când scriem un **SELECT**, noi **NU** doar interogăm o bază de date — **noi învațăm să punem întrebări clare, să filtrăm, să conectăm și să înțelegem REALITATEA.**

Reflecție finală, concluzii generale, Studiu de caz/ Exemplu – continuare

Transformă fiecare interogare SQL într-o întrebare existențială.

Nu întreba doar „Ce temperatură are CPU-ul?” — întreabă:
„Este sistemul meu în echilibru? Ce-mi spune temperatura despre sănătatea lui?”

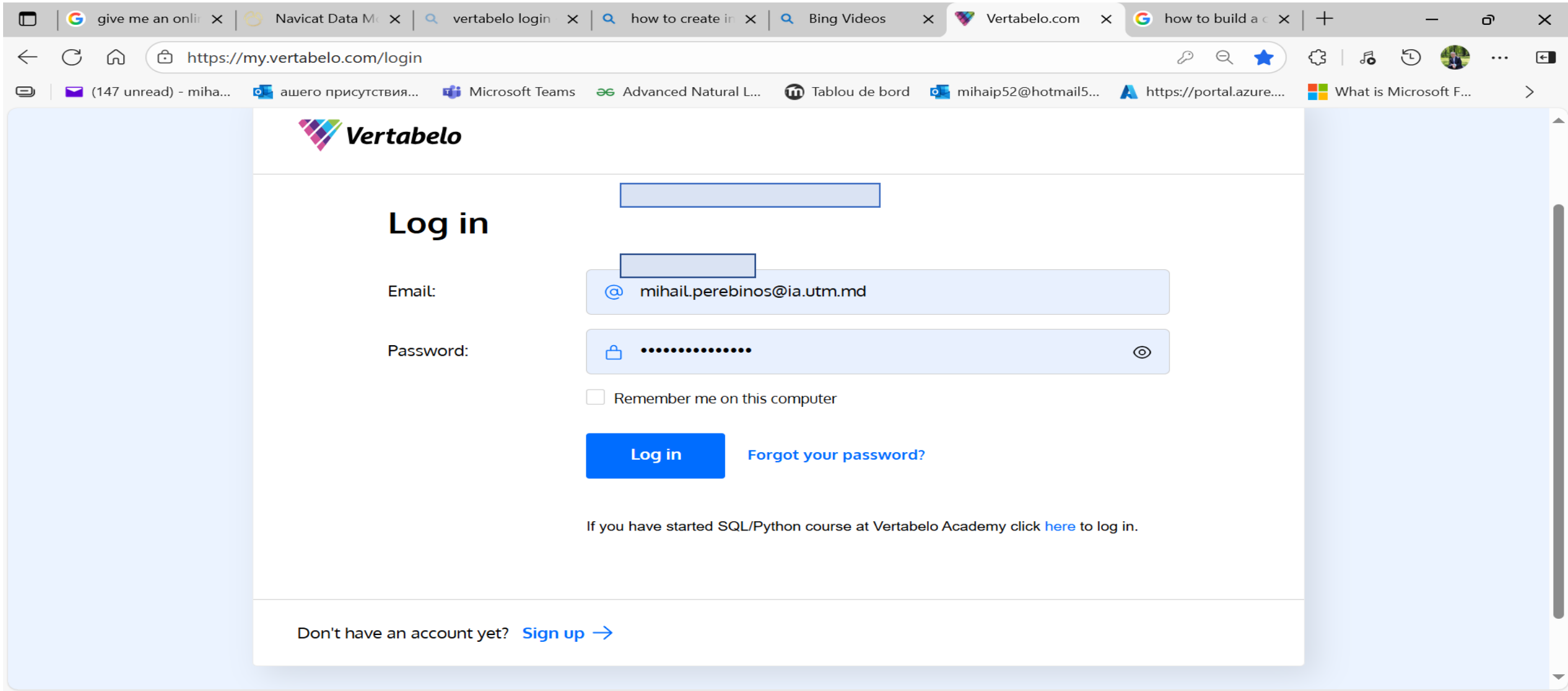
Folosește explicații simple și vizuale:

SELECT = „Ce vreau să aflu?”
WHERE = „Ce condiții pun ca să fie relevant?”
JOIN = „Cum leg realitățile între ele?”

Construiește elemente de reflecție: După fiecare interogare, incercați să răspundeți la următoarele întrebări:
Ce am aflat?
Ce înseamnă asta pentru sistem?
Ce decizie pot lua?

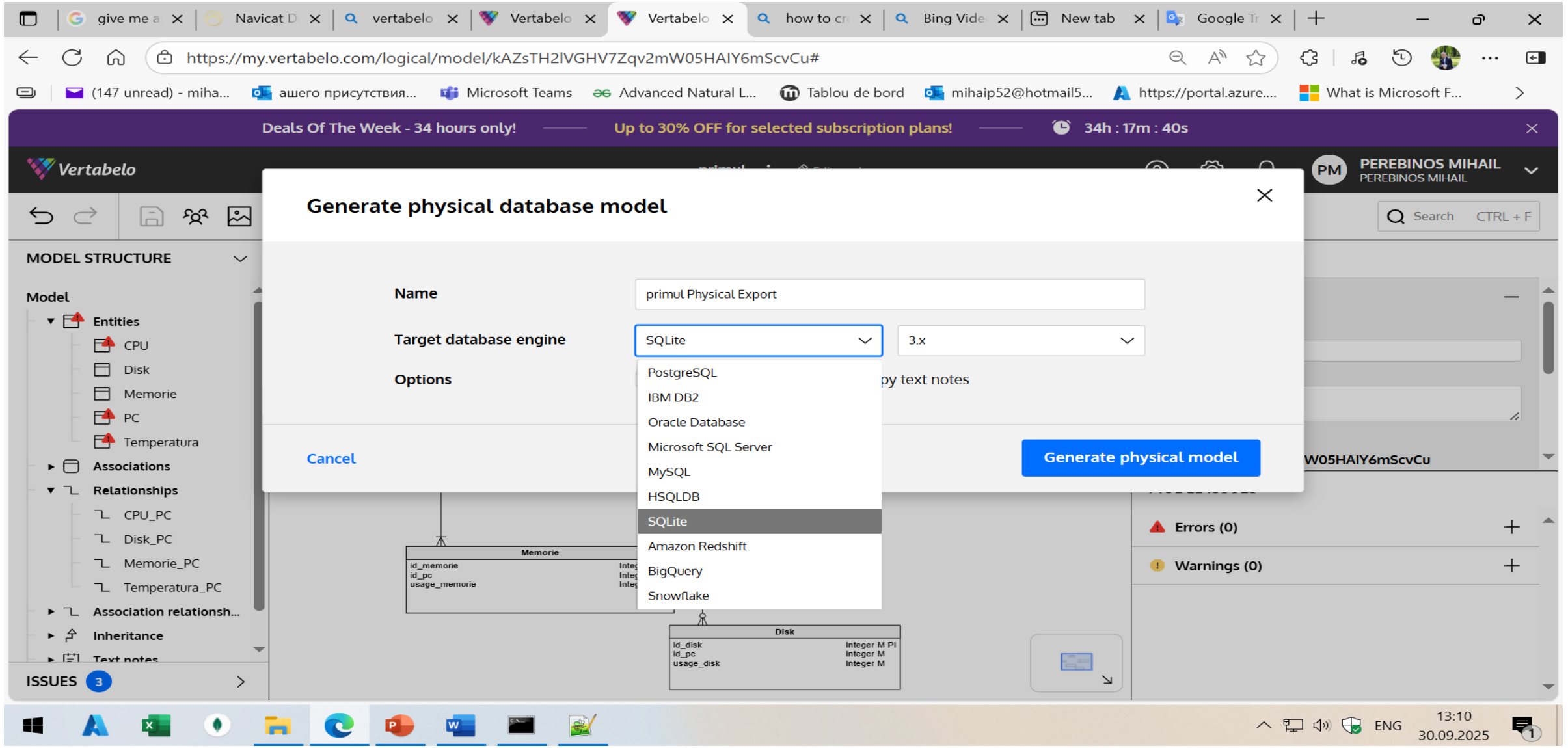
Vertabelo – tool-sul spre structura BD

Studiu de caz/ Exemplu – continuare



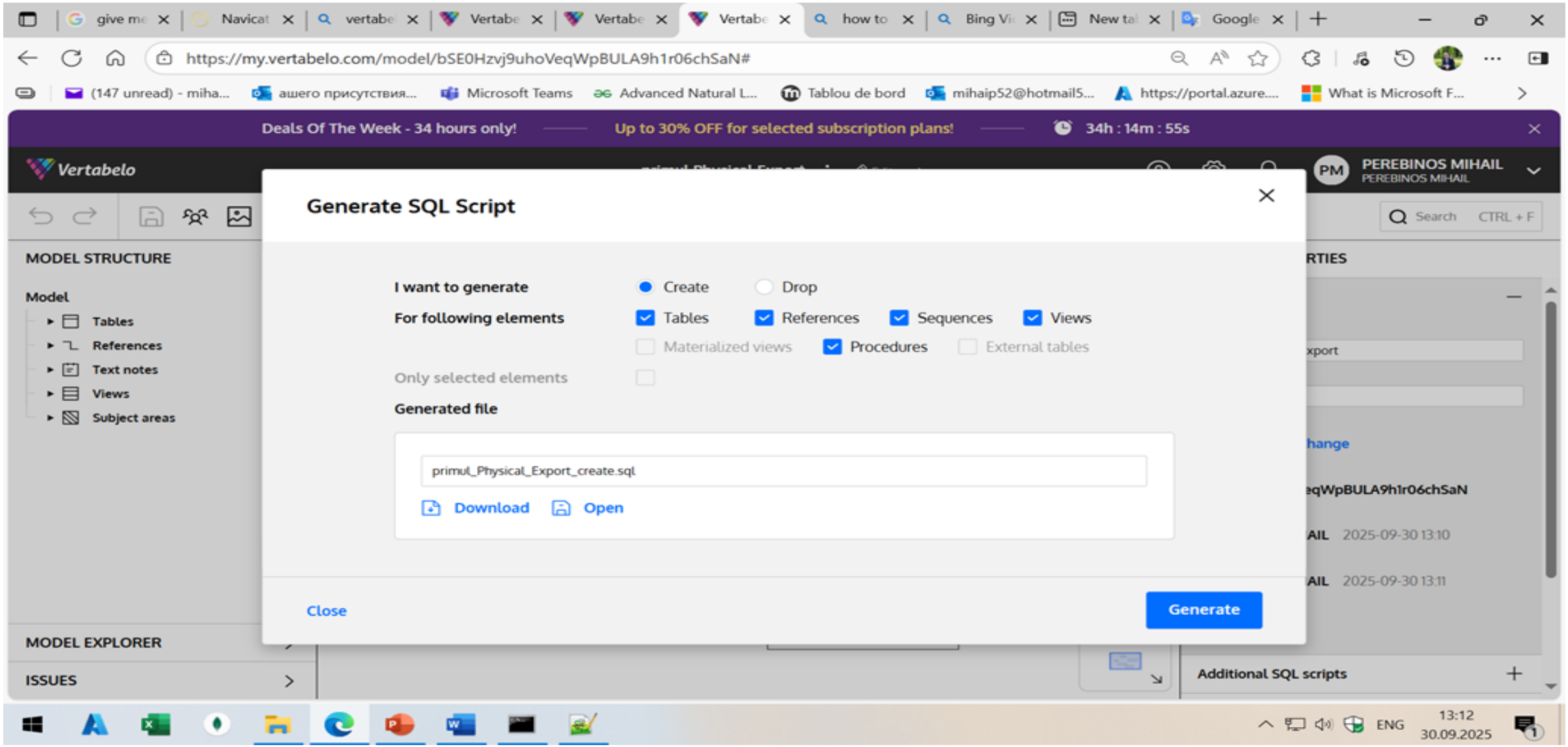
Vertabelo – tool-sul spre structura BD

Studiu de caz/ Exemplu – continuare



Vertabelo – tool-sul spre structura BD

Studiu de caz/ Exemplu – continuare



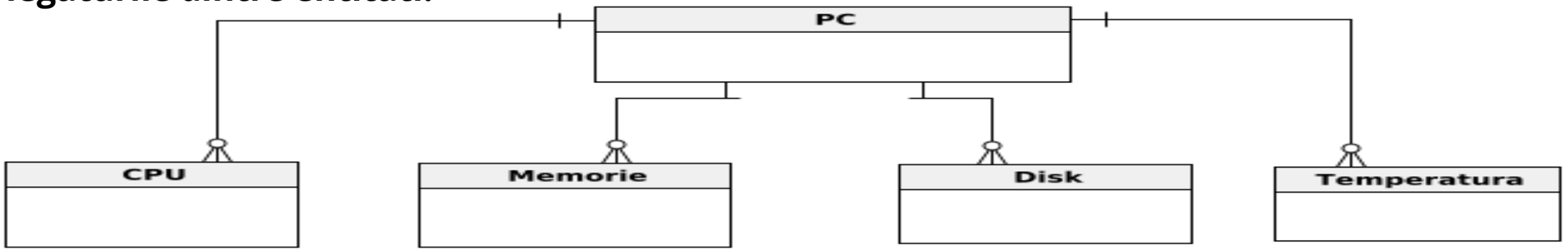
Modelarea datelor – modelul Conceptual,
Studiu de caz/ Exemplu – continuare

Ce este o diagrama entitate-relație?

O diagramă entitate-relație (DER/ERD) este **o reprezentare picturală a informațiilor care pot fi captate de o bază de date**. O astfel de „imagine” servește două scopuri.

Permite profesioniștilor din domeniul bazelor de date să descrie un design general concis, dar precis. **O diagramă ER poate fi ușor transformată într-o schemă relațională**.

Modelul **conceptual** **are ca scop stabilirea entităților si a atributelor ce le descriu precum si legăturile dintre entitati.**

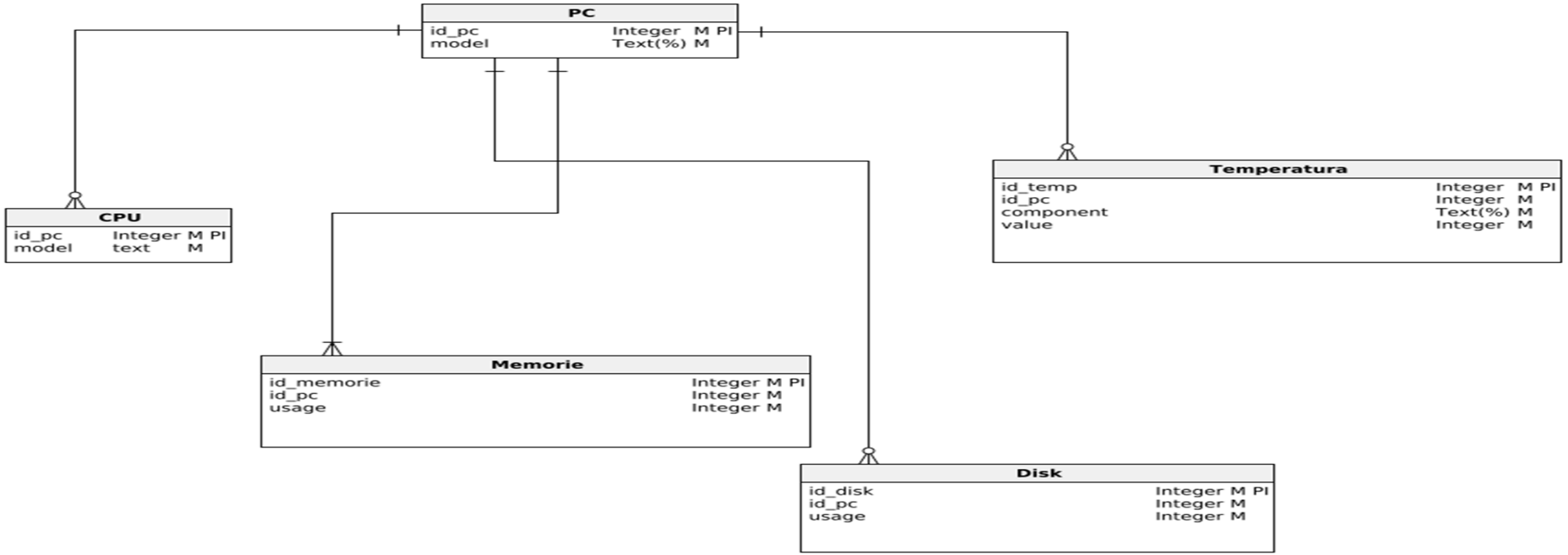


Vertabelo

Modelarea datelor – modelul logic

Studiu de caz/ Exemplu – continuare

1. Modelul **logic de date** definește structura elementelor de date și **stabilește relațiile dintre acestea prin atribute.**



Modelarea datelor – modelul fizic

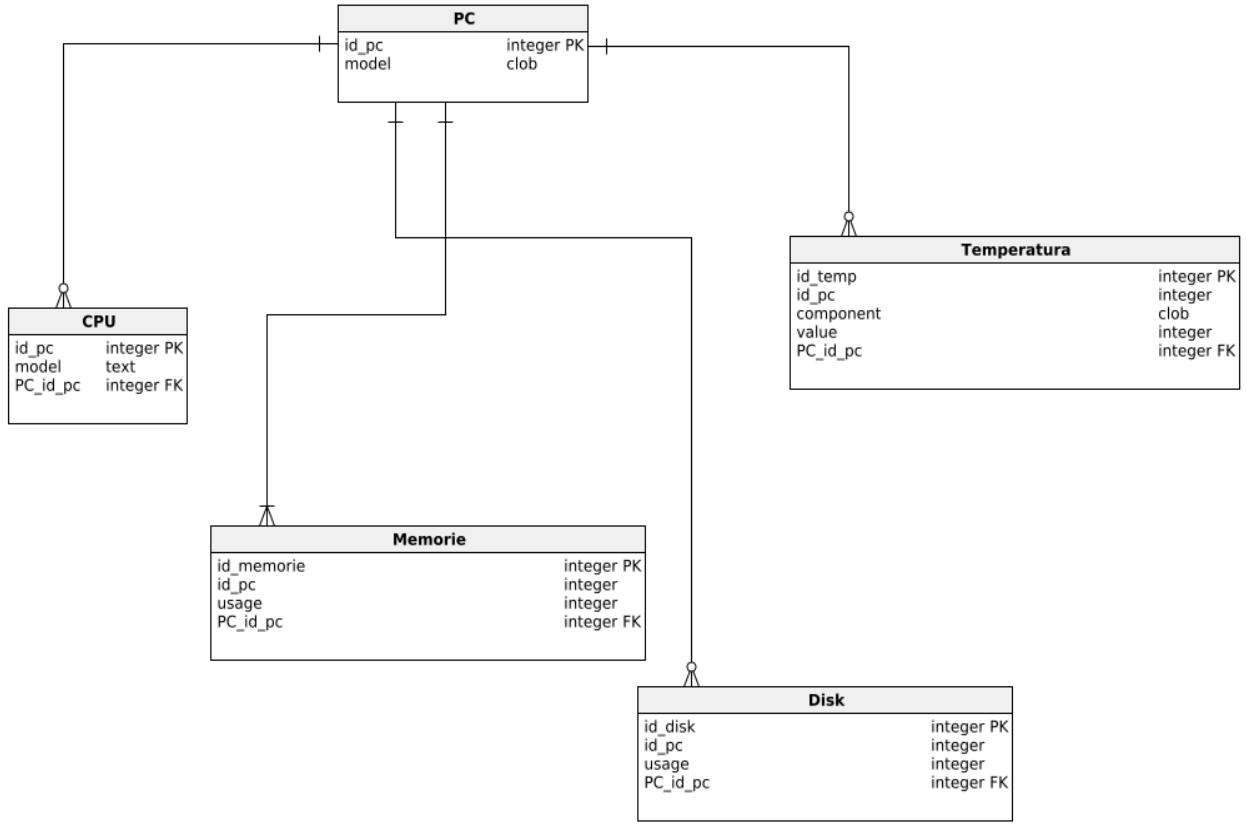
Studiu de caz/ Exemplu – continuare

1. Modelul **fizic de date** **stabileste** modelul de date specific **bazei de date**.

```
CREATE TABLE PC (
  id_pc INTEGER PRIMARY KEY,
  model TEXT
);
CREATE TABLE CPU (
  id_cpu INTEGER PRIMARY KEY,
  id_pc INTEGER,
  usage INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);
CREATE TABLE Memorie (
  id_memorie INTEGER PRIMARY KEY,
  id_pc INTEGER,
  usage INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);
```

```
CREATE TABLE Disk (
  id_disk INTEGER PRIMARY KEY,
  id_pc INTEGER,
  usage INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);

CREATE TABLE Temperatura (
  id_temp INTEGER PRIMARY KEY,
  id_pc INTEGER,
  component TEXT,
  value INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);
```



✨ ***Moment de reflecție*:** Ce înseamnă să transformăm o componentă fizică într-un rând într-o tabelă?

Modulul 2: Crearea bazei de date în SQLite

Studiu de caz/ Exemplu – continuare

CLI – Command-Line Interface, GUI

```
CREATE TABLE PC (
  id_pc INTEGER PRIMARY KEY,
  model TEXT
);
CREATE TABLE CPU (
  id_cpu INTEGER PRIMARY KEY,
  id_pc INTEGER,
  usage INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);
CREATE TABLE Memorie (
  id_memorie INTEGER PRIMARY KEY,
  id_pc INTEGER,
  usage INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);
```

```
CREATE TABLE Disk (
  id_disk INTEGER PRIMARY KEY,
  id_pc INTEGER,
  usage INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);

CREATE TABLE Temperatura (
  id_temp INTEGER PRIMARY KEY,
  id_pc INTEGER,
  component TEXT,
  value INTEGER,
  FOREIGN KEY(id_pc) REFERENCES PC(id_pc)
);
```

```
Error: unknown command or invalid arguments:  "MODE". Enter ".help" for help
sqlite> .mode table
sqlite> .headers on
sqlite> select * from Temperatura;
+---------+-------+-----------+-------+
| id_temp | id_pc | component | value |
+---------+-------+-----------+-------+
| 1       | 1     | CPU       | 88    |
| 2       | 2     | CPU       | 60    |
| 3       | 3     | GPU       | 35    |
| 4       | 1     | GPU       | 75    |
| 5       | 2     | GPU       | 75    |
| 6       | 3     | CPU       | 90    |
+---------+-------+-----------+-------+
sqlite> select * from PC
   ...> Program interrupted.

C:\sqlite>sqlite3 pc_metriks.db
SQLite version 3.50.4 2025-07-30 19:33:53
Enter ".help" for usage hints.
sqlite> select * from PC;
1|Dell XPS 13 (2017)
2|Dell XPS 13 (2018)
3|Dell XPS 13 (2019)
sqlite> .mode table
sqlite> .headers on
sqlite> select * from PC;
+-------+--------------------+
| id_pc |       model        |
+-------+--------------------+
| 1     | Dell XPS 13 (2017) |
| 2     | Dell XPS 13 (2018) |
| 3     | Dell XPS 13 (2019) |
```

Modulul 3: Operații CRUD în SQLite

Studiu de caz/ Exemplu – continuare

Scop: Să învățăm cum se populează și se modifică realitatea digitală.

- **Create:** `INSERT INTO CPU (id_pc, usage) VALUES (1, 45);`
- **Read:** `SELECT * FROM CPU WHERE usage > 80;`
- **Update:** `UPDATE CPU SET usage = 30 WHERE id_cpu = 1;`
- **Delete:** `DELETE FROM CPU WHERE id_cpu = 1;`

✨ ***Moment de reflecție:*** Ce înseamnă să „ștergi” o componentă? Ce rămâne?

Modulul 4: Interogări MONO în CLI SQLite

Studiu de caz/ Exemplu – continuare

Scop: Să înțelegem cum SELECT-ul devine fereastră spre realitate.

Obiectiv 1 – Performanță (temperaturi optime)

Cod Sql

```
SELECT model, component, value
FROM PC
JOIN Temperatura ON PC.id_pc = Temperatura.id_pc
WHERE value BETWEEN 30 AND 70;
```

Obiectiv 2 – Longevitate (resurse sub control)

Cod Sql

```
SELECT model, usage
FROM CPU
JOIN PC ON CPU.id_pc = PC.id_pc
WHERE usage < 50;
```

Modulul 4: Interogări MONO în CLI SQLite

Studiu de caz/ Exemplu – continuare

Scop: Să înțelegem cum SELECT-ul devine fereastră spre realitate.

Obiectiv 3 – Fiabilitate și stabilitate (evitarea blocajelor)

Cod Sql

```
SELECT model, usage
FROM Memorie
JOIN PC ON Memorie.id_pc = PC.id_pc
WHERE usage > 80;
```

Obiectiv 4 – Securitate (spațiu pe disc)

Cod Sql

```
SELECT model, usage
FROM Disk
JOIN PC ON Disk.id_pc = PC.id_pc
WHERE usage > 90;
```

✨ ***Moment de reflecție*:** Ce înseamnă să definești „normalul” printr-un WHERE?

Modulul 5: Scriptul în limbaje de programare Python

Studiu de caz/ Exemplu – continuare

BD SQLITE 'pc_metriks.db'

Tabelul PC

```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from PC;
+-------+--------------------+
| id_pc |       model        |
+-------+--------------------+
| 1     | Dell XPS 13 (2017) |
| 2     | Dell XPS 13 (2018) |
| 3     | Dell XPS 13 (2019) |
+-------+--------------------+
sqlite>
```

Tabelul Temperatura

```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from Temperatura;
+---------+------+-----------+-------+
| id_temp | id_pc | component | value |
+---------+------+-----------+-------+
| 1       | 1     | CPU       | 88    |
| 2       | 2     | CPU       | 60    |
| 3       | 3     | GPU       | 35    |
| 4       | 1     | GPU       | 75    |
| 5       | 2     | GPU       | 75    |
| 6       | 3     | CPU       | 90    |
+---------+------+-----------+-------+
```

```
import sqlite3

conn = sqlite3.connect('pc_metriks.db')
cursor = conn.cursor()

cursor.execute("SELECT a.model, b.component, b.value FROM PC a  "
               "JOIN Temperatura b ON a.id_pc = b.id_pc WHERE b.value > 80")

# Preluarea și afișarea rezultatele

print()
print("Rezultate unde valoarea temperaturii > 80:")

for row in cursor.fetchall():
    model, component, value = row
    print(f"Model: {model}, Component: {component}, Value: {value}")

conn.close()
```

```
D:\PycharmProjects\pythonProject38\.venv\Scripts\python.exe
D:\PycharmProjects\pythonProject38\main.py
Rezultate unde valoarea temperaturii > 80:
Model: Dell XPS 13 (2017), Component: CPU, Value: 88
Model: Dell XPS 13 (2019), Component: CPU, Value: 90
Process finished with exit code 0
```

Modulul 5: Scriptul în limbaje de programare Flask

BD SQLITE 'pc_metriks.db'

Tabelul PC

```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from PC;
+-------+--------------------+
| id_pc |       model        |
+-------+--------------------+
| 1     | Dell XPS 13 (2017) |
| 2     | Dell XPS 13 (2018) |
| 3     | Dell XPS 13 (2019) |
+-------+--------------------+
sqlite>
```

Tabelul Temperatura

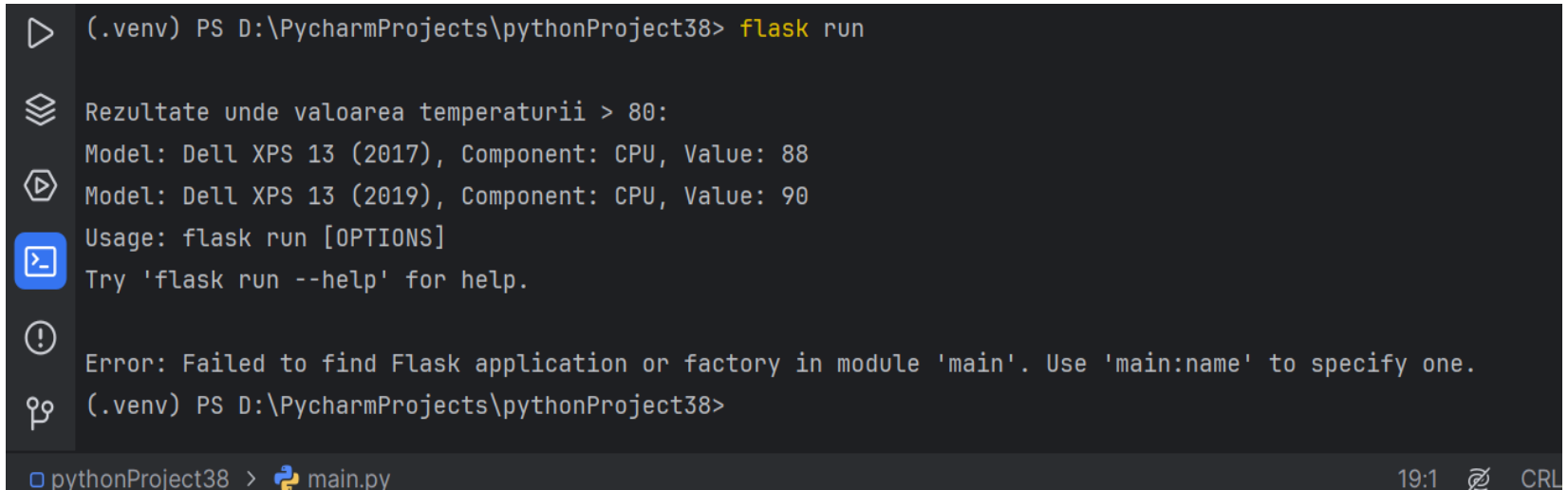
```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from Temperatura;
+---------+------+-----------+-------+
| id_temp | id_pc | component | value |
+---------+------+-----------+-------+
| 1       | 1     | CPU       | 88    |
| 2       | 2     | CPU       | 60    |
| 3       | 3     | GPU       | 35    |
| 4       | 1     | GPU       | 75    |
| 5       | 2     | GPU       | 75    |
| 6       | 3     | CPU       | 90    |
+---------+------+-----------+-------+
```

```python
# Flask
# python
from flask import Flask, render_template
import sqlite3

app = Flask(__name__)

@app.route('/')
def index():
    conn = sqlite3.connect('pc.db')
    cursor = conn.cursor()
    cursor.execute("SELECT model, usage FROM CPU JOIN PC ON
CPU.id_pc = PC.id_pc WHERE usage > 80")
    data = cursor.fetchall()
    conn.close()
    return render_template('index.html', data=data)
```

lansare in Pycharm



Modulul 5: Scriptul în limbaje de programare Streamlit

BD SQLITE 'pc_metriks.db'

Tabelul PC

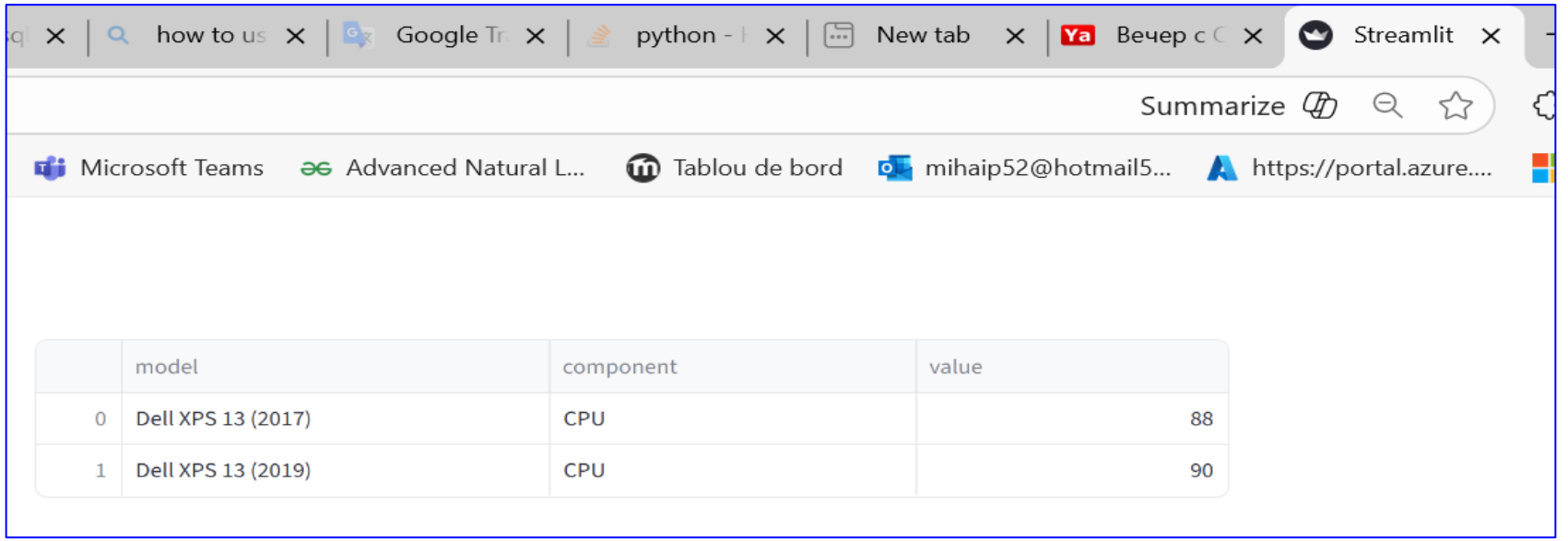
```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from PC;
+-------+--------------------+
| id_pc |       model        |
+-------+--------------------+
| 1     | Dell XPS 13 (2017) |
| 2     | Dell XPS 13 (2018) |
| 3     | Dell XPS 13 (2019) |
+-------+--------------------+
sqlite>
```

Tabelul Temperatura

```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from Temperatura;
+---------+------+-----------+-------+
| id_temp | id_pc | component | value |
+---------+------+-----------+-------+
| 1       | 1     | CPU       | 88    |
| 2       | 2     | CPU       | 60    |
| 3       | 3     | GPU       | 35    |
| 4       | 1     | GPU       | 75    |
| 5       | 2     | GPU       | 75    |
| 6       | 3     | CPU       | 90    |
+---------+------+-----------+-------+
```

```
*python
import streamlit as st
import sqlite3
import pandas as pd

conn = sqlite3.connect('pc.db')
df = pd.read_sql_query("SELECT model, usage FROM CPU JOIN PC ON CPU.id_pc =
PC.id_pc WHERE usage > 80", conn)
st.dataframe(df)
conn.close()
```



	model	component	value
0	Dell XPS 13 (2017)	CPU	88
1	Dell XPS 13 (2019)	CPU	90

Modulul 5: Scriptul în limbaje de programare Tkinter

BD SQLITE 'pc_metriks.db'

Tabelul PC

```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from PC;
+-------+--------------------+
| id_pc |       model        |
+-------+--------------------+
| 1     | Dell XPS 13 (2017) |
| 2     | Dell XPS 13 (2018) |
| 3     | Dell XPS 13 (2019) |
+-------+--------------------+
sqlite>
```

Tabelul Temperatura

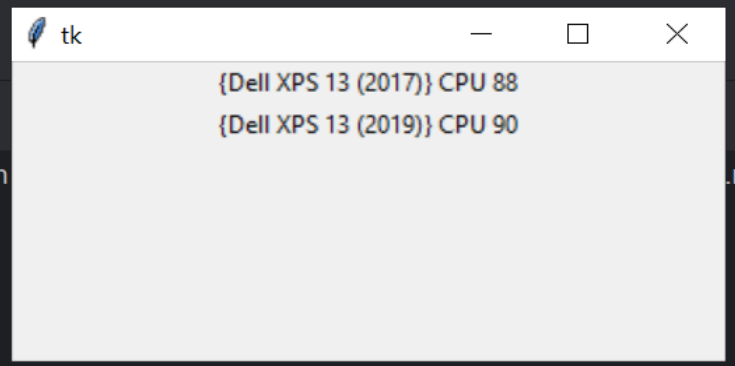
```
sqlite> .mode table
sqlite> .headers on
sqlite> select * from Temperatura;
+---------+------+-----------+-------+
| id_temp | id_pc | component | value |
+---------+------+-----------+-------+
| 1       | 1     | CPU       | 88    |
| 2       | 2     | CPU       | 60    |
| 3       | 3     | GPU       | 35    |
| 4       | 1     | GPU       | 75    |
| 5       | 2     | GPU       | 75    |
| 6       | 3     | CPU       | 90    |
+---------+------+-----------+-------+
```

main.py

```python
# PYTHON
import sqlite3
from tkinter import *

conn = sqlite3.connect('pc_metriks.db')
cursor = conn.cursor()
cursor.execute("SELECT a.model, b.component, b.value FROM PC a  "
               "JOIN Temperatura b ON a.id_pc = b.id_pc WHERE b.value > 80")
rows = cursor.fetchall()
conn.close()

root = Tk()
for row in rows:
    Label(root, text=row).pack()
root.mainloop()
```



Concluzii

- **Fiecare interogare SQL e o întrebare despre REALITATE.**
- **Fiecare tabelă este o metaforă a unei *entitati* ce reflectă REALITATEA modelată.**
- **Fiecare limbaj e o fereastră spre înțelegere a REALITĂȚII.**
- **Fiecare inginer / student / proiectant este un constructor de punți între DATE și SENS.**