L2. SQL DDL. Setarea constrângerilor. Crearea și popularea tabelelor. Importul și exportul datelor.

Pentru aplicațiile mai puțin complexe, schema bazei de date relaționale rezultă direct din analiza cerințelor. În cazul aplicațiilor complexe este necesară proiectarea conceptuală. Proiectarea unei baze de date complexe se va discuta ulterior în cadrul cursul bazei de date, respectiv mai pe larg în cadrul cursului de proiectarea bazelor de date. Structura generica a unei baze de date relaționale este discutată în continuare.

2.1. Modelul relațional

Modelul relațional se bazează pe algebra relațională propusă de E.F.Codd la începutul anilor '70. O bază de date relațională constă dintr-o colecție de relații (sau tabele) asupra cărora se aplică operatori relaționali pentru a gestiona datele. O relație este un set de înregistrări (rânduri). Fiecare rând este un tuplu distinct {a1, a2, … an}. El are un număr fix de atribute (coloane) și fiecare atribut are un anumit tip sau domeniu.

Relația este descrisă de două componente:

- **Instanța**: un tabel fizic cu rânduri și coloane. Fiecare rând reprezintă o înregistrare care denotă o entitate validă din aplicație. Fiecare coloană corespunde unui atribut și poate conține o singură valoare. (nr. de rânduri = cardinalitate, nr. de câmpuri = grad)
- **Schema**: specifică numele relației, plus numele și tipul fiecărui atribut. De exemplu: Student (sid: șir caractere, nume: șir caractere, nota: număr real).

O tabelă poate conține doar înregistrări distincte. Pentru aceasta, fiecare tabelă va declara o **cheie primară** (primary key – **PK)** care va identifica unic înregistrările din tabelă. Ex.: *sid*

(student id – identificator student) este cheia primară pentru tabela Student. Doi studenți diferiți nu pot avea aceiași valoare de *sid*.

O **cheie externă** (foreign key – **FK**) este un set de câmpuri dintr-o tabelă care este folosit pentru a referi o înregistrare din altă tabelă. Ea este ca un pointer logic și va corespunde de obicei cheii primare a celei de-a doua tabele.

2.2. SQL DDL

Partea limbajului SQL care se ocupă cu gestiunea structurii datelor poartă numele de limbaj de definire a datelor (data definition language - DDL). El este o colecție de comenzi utilizate pentru definirea structurilor de date pentru a descrie o schemă de bază de date. SQL DDL permite definirea și modificarea schemei bazei de date prin adăugarea, modificarea sau ștergerea de tabele sau alte obiecte, de exemplu vederi (*Views*) sau indecși.

A. CREATE TABLE

Folosită pentru a crea o nouă relație (tabel) într-o bază de date existentă.

```
CREATE TABLE nume_tabla (
        coloana_1 tip[(dimensiune)] [constrângeri],
        coloana_2 tip[(dimensiune)] [constrângeri],
        ...
        coloana_n tip[(dimensiune)] [constrângeri],
        [,
        CONSTRAINT nume_constrangere {
                CHECK (expresie logică), |
                PRIMARY KEY (atribut/lista atribute),|
                FOREIGN KEY (atribut local)
                        REFERENCES tabela ref. (atribut),
                }
        …
        ]
);
```

Valorile acceptate pentru *tip* sunt dependente de sistem. În cazul Oracle, acceptă (printre altele): CHAR(n), VARCHAR2(n), INT, INTEGER, FLOAT, REAL, NUMBER(p, s), NUMERIC(p, s), DATE, BLOB.

Câmpul constrângeri poate conține: **DEFAULT** valoare implicita, **NOT NULL** sau **UNIQUE**.

B. ALTER TABLE

Folosită pentru a modifica o relație (tabela) într-o bază de date existentă.

ALTER TABLE nume_tabela_vechi
    RENAME TO nume_tabela_nou;

ALTER TABLE tabela
    ADD nume_colana tip [*constrangeri*];

ALTER TABLE tabela
    MODIFY nume_colana tip [constrangeri];

ALTER TABLE tabela
    DROP COLUMN nume_colana;

ALTER TABLE tabela
    RENAME COLUMN nume_vechi TO nume_nou;

ALTER TABLE tabela
    ADD CONSTRAINT nume *constrângeri*
    *(ex. **CHECK, PRIMARY KEY** sau **FOREIGN KEY**)*

ALTER TABLE tabela
    DROP CONSTRAINT nume_constrângere;

ALTER TABLE tabela
    {ENABLE | DISABLE}
    CONSTRAINT nume_constrângere;

C. DROP TABLE

Folosit pentru a șterge o relație (tabelă) existentă dintr-o bază de date.

DROP TABLE tabla [**PURGE**];

În funcție de sistem, este foarte puțin probabil să se poată recupera datele din tabelul original. În Oracle, dacă este specificat **PURGE**, tabelul mai nu poate fi recuperat prin operație de *Rollback* (se va discuta la capitolul de tranzacții – proiectarea bazelor de date).

2.3. Adăugarea/modificarea/ștergerea de date în/din tabele

Instrucțiunea SQL **INSERT** este folosită pentru a insera noi înregistrări într-un tabel. Sunt acceptate mai multe sintaxe:

INSERT INTO tabela **VALUES** (value1, value2, ...);

Sau, doar pentru anumite coloane (obligatoriu PK si cele marcate NOT NULL):

INSERT INTO tabela(col_i, col_j ...) **VALUES** (value_i, value_j, ...);

Pentru a copia date dintr-un tabelă sursă într-o altă tabelă se va folosi:

INSERT INTO tabelă (col1, col2, ...)

SELECT exp1, exp2, ... **FROM** source_table;

Instrucțiunea SQL **UPDATE** este folosită pentru a modifica înregistrări într-un tabel.

UPDATE table

SET column1=value1, column2=value2, ...

[**WHERE** logical_condition];

Dacă lipsește clauza **WHERE** vor fi modificate toate înregistrările din tabelă! Valorile trebuie să corespundă tipurilor de date din fiecare coloană.

Instrucțiunea **DELETE** este folosită pentru a șterge înregistrările selectate dintr-o tabelă.

DELETE FROM tabelă

[**WHERE** condiție logică];

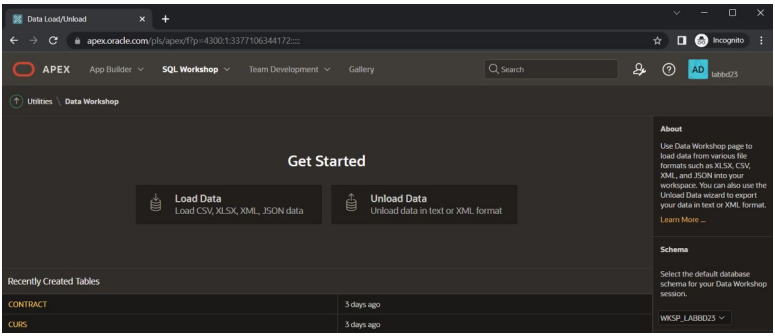
Clauza **WHERE** specifică ce înregistrare sau înregistrări vor fi șterse. Dacă omiteți clauza **WHERE**, toate înregistrările vor fi șterse!

2.4. Importul/exportul datelor folosind APEX SQL Workshop

Date din Oracle XE pot fi exportate folosind interfața Oracle APEX –> SQL Workshop –> Utilities –> Data Workshop –> Unload Data. Exportul se poate face în fișiere CSV, XML sau JSON.

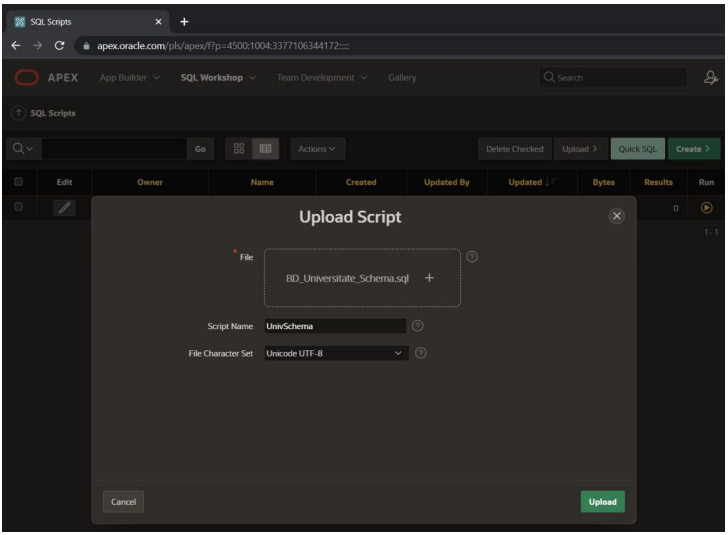
Pentru importul datelor din aceste fișiere se poate folosi

SQL Workshop –> Utilities –> Data Workshop –> Load Data.



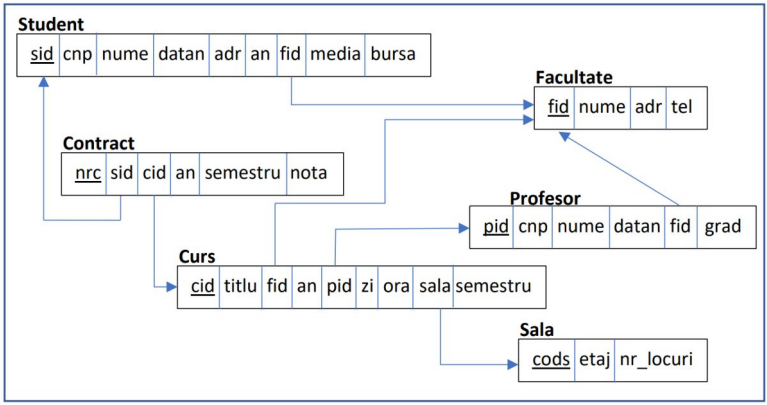
În acest fel pot fi exportate / importante datele din tabelele selectate în momentul exportului.

Pentru exportul structurii tabelelor se poate folosi SQL-Workshop unde selectează tabela si apoi se afișează comanda DDL corespunzătoare în fereastra. Aceasta comanda poate fi apoi copiata intr-un fisier SQL extern care poate fi ulterior rulat ca și Script SQL din SQL Workshop –> SQL Scripts –> Upload apoi RUN.



A. Încărcarea bazei de date pe platforma Oracle Apex

Pentru rularea exercițiilor de laborator se va utiliza schema Universitate prezentată în figura de mai jos.

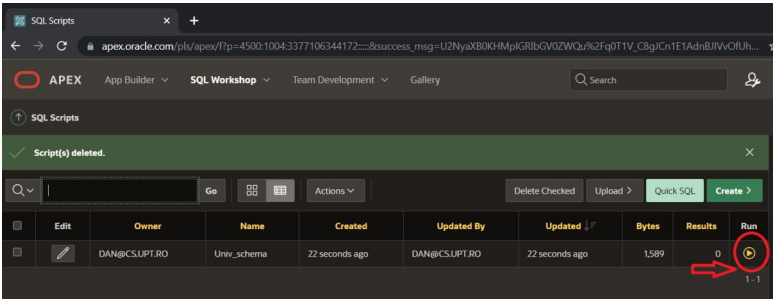


Scurtă descriere:

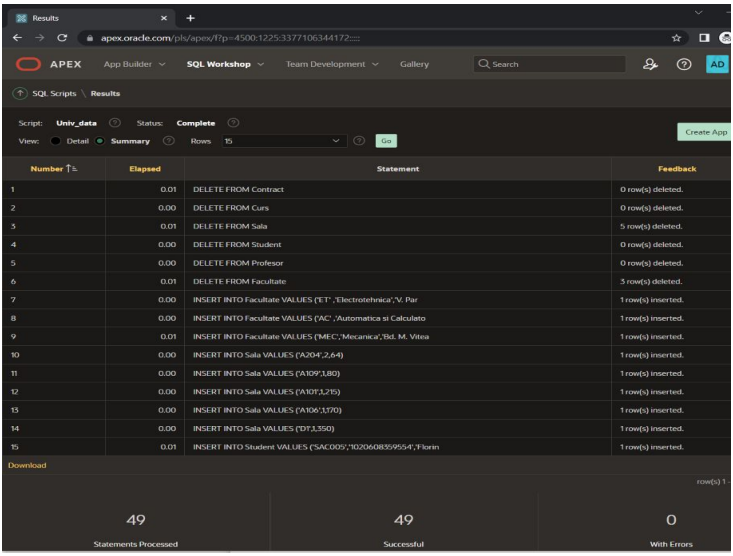
- bursa: se consideră că un student are bursă dacă valoarea bursei este o valoare validă și strict mai mare decât 0
- relația student-curs: un student are un curs, dacă există un contract între student și cursul respectiv
- nume studenți și profesori: numele are formatul, dacă se cer componentele individuale:
  - ‘prenume nume-de-familie’
- CNP-ul are forma ‘GAALLZZXXXXXX’, unde G este codul genului persoanei după notația 1900-1999, ca 1-bărbat și 2-femeie, AA – anul, LL – luna și ZZ – ziua.
- Adresele sunt de forma ‘stradă număr, oraș’

Aceasta este disponibila pe cv.upt.ro si poate fi încărcata în contul APEX propriu în felul următor:

1. De pe campus se salvează (click dreapta + Save As) pe calculatorul propriu fișierele:
    a. BD_Universitate - Shema DDLFile (BD_Universitate_Schema.sql)
    b. BD_Universitate - Continut (date DML)File (BD_Universitate_Data.sql)
2. Din contul de APEX se intra în SQL Workshop – SQL Scripts – Upload.
3. Se selectează de pe disc (Choose file +) fișierul BD_Universitate_Schema.sql, se completează Script Name cu Univ_schema (respectiv Univ_data, la reluare) și se apasă butonul Upload.
4. Se apasă apoi butonul de Run din dreptul liniei corespunzătoare ca în figură, apoi Run Now.

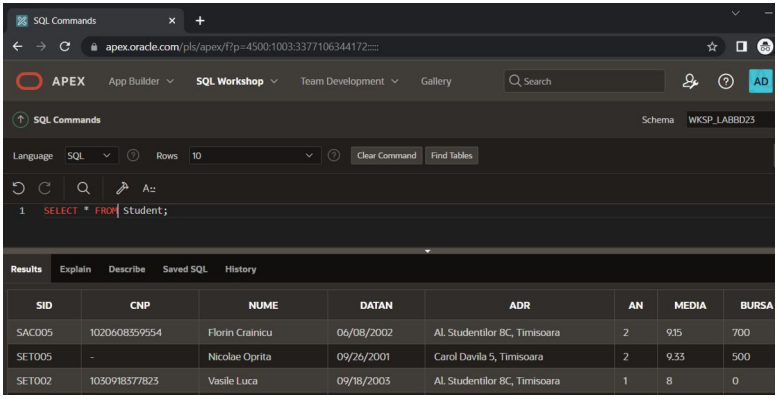


5. Se procedează identic (de la pasul 2) cu fișierul BD_Universitate_Data.sql.
6. După rulare se verifică că nu s-au raportat erori (0 With Errors):



Pentru a verifica vizual datele se va intra în SQL Workshop – SQL Commands si se va rula:

SELECT * FROMStudent;



Resurse externe

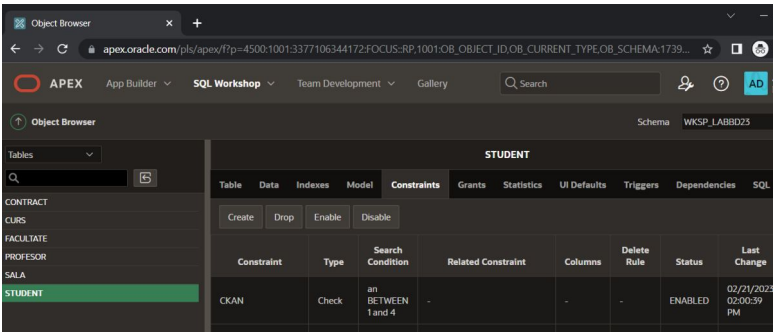
Ref: https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/ALTER-TABLE.html

2.5. Exerciții:

L2.Ex1. Folosind **CREATE TABLE** să se adauge o nouă tabelă ***MAȘINĂ*** în baza de date, cu următoarea strucutră:

Câmp	*Tip*	*Atribute*
Nr Inregistrare	number(5)	Cheie primară
Proprietar	char(6)	Cheie externă, referă Profesor(pid)
Nr Inmatriculare	char(9)	*Notă!* Forma acceptată este „LL CC LLL”, CC nu poate fi 00.
Culoare	varchar2(10)	NOT NULL
Vporbagaj	number(5,2)	NOT NULL, *Restricție*: nu poate depăși 300 de litri și nu poate fi negativ
NrKm	number(9,2)	NOT NULL, default value 0, nu poate fi negativ
AnFabricatie	number(4)	NOT NULL

L2.Ex2. Folosind **ALTER TABLE** să se adauge o constrângere (cu numele MS_ANF_CK) la tabela ***MAȘINĂ*** care să verifice că anul de fabricație este între 1980 și 2024. Se va verifica apoi că această constrângere a fost adăugată prin afișarea constrângerilor din tabela ***MAȘINĂ*** în SQL – Workshop. Pentru aceasta, accesați O*bject Browser*, selectați tabela ***MAȘINĂ*** și accesați fereastra *Constraints*.



L2.Ex3. Folosind comanda **INSERT** să se adauge 3 mașini noi:

- mașina cu numărul *19082*, aparținând profesorului Elisa Zamfirescu, cu numărul de înmatriculare *TM 01 ABC*, fabricată în *2021*, având culoarea roșie, *100* litri volumul portbagajului, *12000* de km la bord

- mașina cu numărul 23*062*, aparținând profesorului Veronica Micle, cu numărul de înmatriculare *TM 02 ABD*, fabricată în *2021*, având culoarea roșie, *100* litri volumul portbagajului, *19000* de km la bord

- mașina cu numărul *33912*, aparținând profesorului Luca Caragiale, cu numărul de înmatriculare *TM 40 PPT*, fabricată în *2023*, având culoarea roșie, *100* litri volumul portbagajului, *8000* de km la bord

L2.Ex4. Folosind comanda **UPDATE** cu **WHERE** să se modifice numărul de kilometri de mașina 33912 la 34000 km.

L2.Ex5. În anul precedent, toate mașinile au parcurs o distanta de 1250 de kilometrii. Folosind comanda **UPDATE**, să se modifice informația din tabele pentru a reflecta acest lucru.

L2.Ex6. Folosind comanda **DELETE** cu **WHERE** să se șteargă mașinile care au peste 20000 de kilometrii.

L2.Ex7. Folosind comanda **DROP**, să se elimine tabela **MAȘINĂ** din schema curentă.

Obs: pentru reluarea rulării unei comenzi SQL scrise anterior (ex. la predarea rezolvărilor) se va merge la SQL Workshop – SQL Commands pe tabul History: