Indicativ_echipa: SIA_16

Studenti: Axinia Anton Mihai

David Alin Mihai

Rusu Ioana-Diana

TEMA: L1. Proiectarea studiului de caz: Gestionarea unei bibliotecii

DS_1: PostgreSQL – Gestiunea cărţilor din bibliotecă

Tip sursă de date

- Model de date: relaţional
- Format de acces: SQL

Descriere structură de date

- Tabele principale: carti, editura, categorie_literatura, cotacarte, persoana, carti_biblioteca
- Câmpuri importante:
  - carti – isbn, titlu, id_editura, id_categorie_literatura, data_publicare
  - editura – id_editura, nume_editura
  - persoana – idpersoana, nume, prenume, cnp, email
- Legături între structuri:
  - carti.id_editura → editura.id_editura
  - carti.id_categorie_literatura → categorie_literatura.id_categorie_literatura
  - carti_biblioteca.isbn → carti.isbn

Implementare sursă de date externă

- Script: tabele dbreaver (1).txt
- Tip fişier: SQL
- Conţinut: comenzi DDL + DML pentru creare şi populare bazei de date

DS_2: Oracle – Gestiunea abonamentelor şi împrumuturilor

Tip sursă de date

- Model de date: relaţional
- Format de acces: SQL (Oracle)

Descriere structură de date

- Tabele: abonament_biblioteca, status_carte, imprumuturi_carti
- Câmpuri importante:

    - abonament_biblioteca – id_abonament, idpersoana, id_biblioteca, data_creare_abonament, data_finalizare_abonament
    - imprumuturi_carti – id_imprumut, id_abonament, id_carte_biblioteca, data_imprumut, data_limita_returnare, data_returnare, id_status
- Legături:
    - imprumuturi_carti.id_abonament → abonament_biblioteca.id_abonament
    - imprumuturi_carti.id_status → status_carte.id_status

Implementare sursă de date externă

- Script: table abonamente oracle (2).txt
- Tip fişier: SQL Oracle
- Conţinut: comenzi DDL + DML pentru structură şi date

DS_3: Neo4j – Reţea geografică pentru localizarea bibliotecilor

Tip sursă de date

- Model de date: graf
- Format de acces: Cypher (Neo4j)

Descriere structură de date

- Noduri: Tara, Judet, Oras, Adresa
- Atribute:
    - Tara – idtara, denumiretara
    - Judet – idjudet, denumirejudet
    - Oras – idoras, denumireoras
    - Adresa – idadresa, strada, numar
- Legături între structuri:
    - Judet → Tara prin relaţia APARTINE
    - Oras → Judet prin relaţia APARTINE
    - Adresa → Oras prin relaţia SE_AFLa_IN

Implementare sursă de date externă

- Script: table neo4j.txt
- Tip fişier: Cypher script (Neo4j)
- Conţinut: creare noduri şi relaţii, ştergere şi reiniţializare bază graf

DS_4: CSV – Date despre personalul bibliotecii

Tip sursă de date

- Model de date: tabelar
- Format de acces: CSV

Descriere structură de date

- Coloane: Nume, Prenume, Functie, Email
- Fiecare rând corespunde unui angajat al bibliotecii
- Nu există legături explicite cu alte surse de date, dar poate fi integrat cu tabelele din PostgreSQL pentru atribuirea responsabililor

Implementare sursă de date externă

- Fișier: personal_biblioteca.csv
- Tip fișier: CSV
- Conținut: date separate prin virgulă, ușor de importat în PostgreSQL sau Excel

IMPLEMENTARE ACCESS surse de date externe

DS_1 – PostgreSQL (REST – PostgREST)

- Tip sursă: REST JSON
- Metodă de acces: HTTP cu HTTPURITYPE + JSON_TABLE
- Integrare: vizualizări Oracle din endpoint-uri REST (ex. /carti, /persoana)

DS_2 – Oracle (DB LINK)

- Tip sursă: Oracle remote
- Metodă de acces: DATABASE LINK + CREATE VIEW din tabele remote

DS_3 – Neo4j (REST API + JSON_TABLE)

- Tip sursă: Neo4j Graph
- Metodă de acces: funcție PL/SQL + UTL_HTTP + JSON_TABLE
- Integrare: view-uri construite pe baza interogărilor Cypher + REST

DS_4 – CSV (External Table Oracle)

- Tip sursă: Fișier .csv

- Metodă de acces: ORGANIZATION EXTERNAL cu ORACLE_LOADER
- Integrare: citit direct de Oracle ca tabel extern

TEMA: L3. Analytical Integration Model: Implementare OLAP views

```sql
CREATE OR REPLACE VIEW max_imprumut_categorie AS
WITH categorii_imprumuturi AS (
    SELECT cl.nume_categorie_literatura,
           COUNT(ic.id_imprumut) AS total_imprumuturi
    FROM imprumuturi ic
    JOIN carti_biblioteca_view cb ON ic.id_carte_biblioteca = cb.id_carte_biblioteca
    JOIN carti_view c ON cb.isbn = c.isbn
    JOIN categorie_literatura_view cl ON c.id_categorie_literatura = cl.id_categorie_literatura
    GROUP BY cl.nume_categorie_literatura
),
max_imprumuturi AS (
    SELECT MAX(total_imprumuturi) AS valoare_maxima
    FROM categorii_imprumuturi
)
SELECT ci.*
FROM categorii_imprumuturi ci
JOIN max_imprumuturi mi ON ci.total_imprumuturi = mi.valoare_maxima;

SELECT * FROM max_imprumut_categorie;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.048 seconds

	NUME_CATEGORIE_LITERATURA	TOTAL_IMPRUMUTURI
1	ArtÃ□	3
2	Tehnologie	3

Worksheet | Query Builder

```sql
CREATE OR REPLACE VIEW statistica_imprumuturi_view AS
SELECT
    CASE
        WHEN cl.nume_categorie_literatura IS NULL THEN 'Total General'
        WHEN e.nume_editura IS NULL THEN 'Subtotal ' || cl.nume_categorie_literatura
        ELSE cl.nume_categorie_literatura
    END AS "Categorie Literară",

    CASE
        WHEN e.nume_editura IS NOT NULL THEN e.nume_editura
        WHEN cl.nume_categorie_literatura IS NULL THEN 'Total General'
        ELSE ' '
    END AS "Editură",

    COUNT(ic.id_imprumut) AS "Număr Cărți Împrumutate"
FROM imprumuturi ic
INNER JOIN carti_biblioteca_view cb ON ic.id_carte_biblioteca = cb.id_carte_biblioteca
INNER JOIN carti_view c ON cb.isbn = c.isbn
INNER JOIN categorie_literatura_view cl ON c.id_categorie_literatura = cl.id_categorie_literatura
INNER JOIN editura_view e ON c.id_editura = e.id_editura
GROUP BY ROLLUP(cl.nume_categorie_literatura, e.nume_editura)
ORDER BY
    cl.nume_categorie_literatura,
    e.nume_editura;


SELECT * FROM statistica_imprumuturi_view;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 21 in 0.107 seconds

	Categorie Literară	Editură	Număr Cărți Împrumutate
1	ArtÃ□	Paralela 45	3
2	Subtotal ArtÃ□		3
3	Biografie	Trei	1
4	Subtotal Biografie		1
5	Fantasy	RAO	1
6	Subtotal Fantasy		1
7	FicÈ□iune	Polirom	1
8	Subtotal FicÈ□iune		1
9	Horror	Litera	1
10	Subtotal Horror		1
11	Istorie	Nemira	1
12	Subtotal Istorie		1
13	Poezie	Curtea Veche	1
14	Subtotal Poezie		1
15	Psihologie	Art	1
16	Subtotal Psihologie		1
17	Tehnologie	Corint	3
18	Subtotal Tehnologie		3
19	È□tiinÈ□Ã□	Humanitas	1
20	Subtotal È□tiinÈ□Ã□		1
21	Total General	Total General	14

```sql
CREATE OR REPLACE VIEW top_imprumut_persoana AS
SELECT nume, prenume, titlu, nr_imprumuturi
FROM (
    SELECT
        p.nume,
        p.prenume,
        c.titlu,
        COUNT(*) AS nr_imprumuturi,
        ROW_NUMBER() OVER (PARTITION BY p.idpersoana ORDER BY COUNT(*) DESC) AS rn
    FROM imprumuturi i
    JOIN persoana_view p ON p.idpersoana = p.idpersoana
    JOIN carti_biblioteca_view cb ON i.id_carte_biblioteca = cb.id_carte_biblioteca
    JOIN carti_view c ON cb.isbn = c.isbn
    GROUP BY p.idpersoana, p.nume, p.prenume, c.titlu
) sub
WHERE rn = 1
ORDER BY nume, prenume;


SELECT * FROM top_imprumut_persoana;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.038 seconds

	NUME	PRENUME	TITLU	NR_IMPRUMUTURI
1	Dumitru	Paul	Ghidul Picturii	3
2	Georgescu	Andrei	Bazele ProgramÄ□rii	3
3	Ilie	Sorina	Ghidul Picturii	3
4	Ionescu	Maria	Bazele ProgramÄ□rii	3
5	Marin	Elena	Ghidul Picturii	3
6	Popescu	Ion	Ghidul Picturii	3
7	Radu	Cristina	Bazele ProgramÄ□rii	3
8	Stan	Ioana	Ghidul Picturii	3
9	Toma	Florin	Bazele ProgramÄ□rii	3
10	Vasile	Mihai	Bazele ProgramÄ□rii	3

TEMA: P4. REST and Web Model

POSTGRES LINK VIEW

```
SELECT HTTPURITYPE.createuri('http://localhost:3000/carti').getclob() as doc
from dual;


with json as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/persoana')
   .getclob() as doc from dual)
SELECT
  idpersoana, nume, prenume, cnp, email as persoana
FROM  JSON_TABLE( (select doc from json) , '$[*]'
          COLUMNS (
             idpersoana           PATH '$.idpersoana'
             , nume             PATH '$.nume'
             , prenume          PATH '$.prenume'
             , cnp              PATH '$.cnp'
             , email            PATH '$.email'
             )
);
--------------------------------------------------------------------------------


-- 1. Deploy PostgrREST Service (localhost:3000)
-- 2. Check Oracle ACL Security settings on REST host and port
-- 3. Connect to HTTP with Oracle 21c XE
--------------------------------------------------------------------------------------------------------------
--------------
---- Create REST REMOTE Views: Postgres Data Source
--------------------------------------------------------------------------------------------------------------
--------------
--- Direct Data Source Access ------------------------------------------------------------------------
---------------------
```

```
CREATE OR REPLACE VIEW persoana_view AS
with rest_doc as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/persoana')
   .getclob() as doc from dual)
SELECT
  idpersoana, nume, prenume, cnp, email as persoana
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
          COLUMNS (
             idpersoana           PATH '$.idpersoana'
             , nume             PATH '$.nume'
             , prenume          PATH '$.prenume'
             , cnp              PATH '$.cnp'
             , email            PATH '$.email'
             )
);
---
SELECT * FROM persoana_view;



--------------------------------------------------------------------------------------------------------------
---------------
CREATE OR REPLACE VIEW carti_view AS
with rest_doc as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/carti')
   .getclob() as doc from dual)
SELECT *
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
          columns (
             isbn              varchar(13)    path '$.isbn'
             , titlu         varchar(255) path '$.titlu'
             , idcotacarte                  int path '$.idcotacarte'
```

```
            , id_editura            int path '$.id_editura'
            , id_categorie_literatura           int path '$.id_categorie_literatura'
            , data_publicare            date path '$.data_publicare' )
) ;
---
SELECT * FROM carti_view;


-------------------------------------------------------------------------------------------------------------
--------------
CREATE OR REPLACE VIEW editura_view AS
with rest_doc as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/editura')
   .getclob() as doc from dual)
SELECT *
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
          columns (
             id_editura               number(4)      path '$.id_editura'
             ,nume_editura               varchar(100)   path '$.nume_editura'
             )
);
---
SELECT * FROM editura_view;
-------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE VIEW categorie_literatura_view AS
with rest_doc as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/categorie_literatura')
   .getclob() as doc from dual)
SELECT *
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
          columns (
             id_categorie_literatura               number(4)      path '$.id_categorie_literatura'
```

```
            ,nume_categorie_literatura              varchar(100)   path
'$.nume_categorie_literatura'
            )
);
---
SELECT * FROM categorie_literatura_view;
-----------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE VIEW cotacarte_view AS
with rest_doc as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/cotacarte')
   .getclob() as doc from dual)
SELECT *
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
         columns (
            idcotacarte              number(4)      path '$.idcotacarte'
            ,starecota              varchar(50)   path '$.starecota'
            )
);
---
SELECT * FROM cotacarte_view;
-----------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE VIEW cotacarte_view AS
with rest_doc as
   (SELECT HTTPURITYPE.createuri('http://localhost:3000/cotacarte')
   .getclob() as doc from dual)
SELECT *
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
         columns (
            idcotacarte              number(4)      path '$.idcotacarte'
            ,starecota              varchar(50)   path '$.starecota'
            )
```

```
);
---
SELECT * FROM cotacarte_view;

CREATE OR REPLACE VIEW carti_biblioteca_view AS
WITH rest_doc AS (
   SELECT HTTPURITYPE.createuri('http://localhost:3000/carti_biblioteca')
   .getclob() AS doc FROM dual
)
SELECT *
FROM JSON_TABLE(
   (SELECT doc FROM rest_doc), '$[*]'
   COLUMNS (
      id_carte_biblioteca NUMBER(4) PATH '$.id_carte_biblioteca',
      id_biblioteca       NUMBER(4) PATH '$.id_biblioteca',
      isbn                VARCHAR2(13) PATH '$.isbn',
      acces_digital       VARCHAR2(5) PATH '$.acces_digital', -- Stocăm 'true'/'false'
      mediatip            VARCHAR2(50) PATH '$.mediatip'
   )
);
SELECT * FROM carti_biblioteca_view;
```

ORACLE LINK VIEW

```
--- Oracle DB Link to Oracle Database Schema
ROLLBACK;
ALTER SESSION CLOSE DATABASE LINK abonamenteDB;
DROP DATABASE LINK abonamenteDB;
CREATE DATABASE LINK abonamenteDB
  CONNECT TO abonamente IDENTIFIED BY abonamente
  USING '//localhost:1521/XEPDB1';
```

```
select * from user_db_links;

--- Check DB_LINK

select * from user_tables@abonamenteDB;

select * from abonament_biblioteca@abonamenteDB;

--- Create views on remote tables

DROP VIEW ABONAMENTE_VIEW;

CREATE OR REPLACE VIEW ABONAMENTE_VIEW AS

SELECT id_abonament,idpersoana, id_biblioteca, data_creare_abonament,
data_finalizare_abonament

FROM abonament_biblioteca@abonamenteDB;

select * from ABONAMENTE_VIEW;

--

DROP VIEW STATUS_VIEW;

CREATE OR REPLACE VIEW STATUS_VIEW AS

SELECT id_status, status

FROM status_carte@abonamenteDB;

select * from STATUS_VIEW;

--

DROP VIEW IMPRUMUTURI;

CREATE OR REPLACE VIEW IMPRUMUTURI AS

SELECT id_imprumut, id_abonament, id_carte_biblioteca, data_imprumut,
data_limita_returnare, data_returnare, id_status

FROM imprumuturi_carti@abonamenteDB;

select * from IMPRUMUTURI;
```

NEO4J LINK VIEW

```
-- 28_AM_JSON_Neo4J_View.sql

--------------------------------------------------------------------------------

-- CONNECT with FDBO on XEPDB1 ---------------------------------------------

--------------------------------------------------------------------------------
```

```
-- 1. Deploy Neo4J to Query the database POST http://localhost:7474/db/neo4j/tx/commit
-- 2. Check Oracle ACL Security settings on REST host and port
-- 3. Connect to HTTP with Oracle 21c
CREATE OR REPLACE FUNCTION query_neo4j_rest_graph_data(
   REST_URL VARCHAR2,
   CYPHER_QUERY VARCHAR2,
   USER_NAME VARCHAR2,
   PASS VARCHAR2
   )
RETURN clob IS
 l_req   utl_http.req;
 l_resp  utl_http.resp;
 l_buffer clob;
 l_UserPass VARCHAR2(2000) := USER_NAME || ':' || PASS;
 l_Statement VARCHAR2(4000) :=
   REPLACE('{ "statements": [ { "statement": "$statement" } ] }', '$statement',
CYPHER_QUERY);
begin
 dbms_output.put_line(l_Statement);
 l_req  := utl_http.begin_request(REST_URL, 'POST');
 -- utl_http.set_header(l_req, 'user-agent', 'mozilla/4.0');
 utl_http.set_header(l_req, 'Content-type', 'application/json');
 utl_http.set_header(l_req, 'Content-Length', length(l_Statement));
 utl_http.set_body_charset('UTF-8');
 UTL_HTTP.set_header(l_req, 'Authorization', 'Basic ' ||

UTL_RAW.cast_to_varchar2(UTL_ENCODE.base64_encode(UTL_I18N.string_to_raw(l_Us
erPass, 'AL32UTF8'))));


 utl_http.WRITE_TEXT (l_req, l_Statement);
 -- utl_http.WRITE_RAW (l_req, UTL_RAW.CAST_TO_RAW(l_Statement));
```

```
  l_resp := utl_http.get_response(l_req);
  UTL_HTTP.READ_TEXT(l_resp, l_buffer);
  utl_http.end_response(l_resp);
  return l_buffer;
end;
/
begin
dbms_output.put_line(
    query_neo4j_rest_graph_data(
    'http://localhost:7474/db/neo4j/tx/commit',
    'MATCH (city:City) RETURN *',
    'neo4j', 'administrator')
);
end;
/
--curl http://admin:secret@localhost:8080/mds/Locations
SELECT query_neo4j_rest_graph_data(
    'http://localhost:7474/db/neo4j/tx/commit',
    'MATCH (oras:Oras) RETURN *',
    'neo4j', 'administrator')
from dual;
--------------------------------------------------------------------------------
WITH json AS (
    SELECT query_neo4j_rest_graph_data(
        'http://localhost:7474/db/neo4j/tx/commit',
        'MATCH (oras:Oras) RETURN *',
        'neo4j', 'administrator') doc
    FROM dual
)
SELECT
```

```
  jt.idOras,
  jt.denumireOras
FROM JSON_TABLE(
    (SELECT doc FROM json),
    '$.results[*].data[*].row[*]'  -- Accesăm corect array-ul cu rezultate
    COLUMNS (
        idOras         PATH '$[0].idoras' NULL ON ERROR,  -- idOras este primul element din array-ul "row"
        denumireOras   PATH '$[0].denumireoras' NULL ON ERROR   -- denumireOras este al doilea element din array-ul "row"
    )
) jt;
--------------------------------------------------------------------------------
CREATE OR REPLACE VIEW oras_view_neo4j AS
WITH json AS (
    SELECT query_neo4j_rest_graph_data(
        'http://localhost:7474/db/neo4j/tx/commit',
        'MATCH (oras:Oras)-[:APARTINE]->(judet:Judet) RETURN judet, oras',
        'neo4j', 'administrator') doc
    FROM dual
)
SELECT
  jt.idOras,
  jt.denumireOras,
  jt.idJudet  -- Adăugăm idJudet ca referință la județ
FROM JSON_TABLE(
    (SELECT doc FROM json),
    '$.results[*].data[*].row'  -- Selectăm direct array-ul "row"
    COLUMNS (
        idJudet       PATH '$[0].idjudet' NULL ON ERROR,  -- idJudet din primul obiect
        idOras        PATH '$[1].idoras' NULL ON ERROR,  -- idOras din al doilea obiect
```

```
        denumireOras  PATH '$[1].denumireoras' NULL ON ERROR  -- denumireOras din al doilea obiect
    )
) jt;


---
SELECT * FROM oras_view_neo4j;
--------------------------------------------------------------------------------
CREATE OR REPLACE VIEW adrese_view_neo4j AS
WITH json AS (
    SELECT query_neo4j_rest_graph_data(
        'http://localhost:7474/db/neo4j/tx/commit',
        'MATCH (adresa:Adresa)-[:SE_AFLa_IN]->(oras:Oras) RETURN adresa, oras',
        'neo4j', 'administrator') doc
    FROM dual
)
SELECT
  jt.idAdresa,
  jt.strada,
  jt.numar,
  jt.idOras  -- Adăugăm idOras (referință la orașul părinte)
FROM JSON_TABLE(
    (SELECT doc FROM json),
    '$.results[*].data[*].row'  -- Selectăm direct array-ul "row"
    COLUMNS (
        idAdresa      PATH '$[0].idadresa' NULL ON ERROR,  -- idAdresa este primul element din array-ul "row"
        strada        PATH '$[0].strada' NULL ON ERROR,    -- strada este al doilea element din array-ul "row"
        numar         PATH '$[0].numar' NULL ON ERROR,     -- numar este al treilea element din array-ul "row"
```

```
    idOras      PATH '$[1].idoras' NULL ON ERROR     -- idOras este al patrulea element din array-ul "row"
  )
) jt;


---
SELECT * FROM adrese_view_neo4j;
-------------------------------------------------------------------------------
CREATE OR REPLACE VIEW judete_view_neo4j AS
WITH json AS (
  SELECT query_neo4j_rest_graph_data(
    'http://localhost:7474/db/neo4j/tx/commit',
    'MATCH (judet:Judet)-[:APARTINE]->(tara:Tara) RETURN judet, tara',
    'neo4j', 'administrator') doc
  FROM dual
)
SELECT
 jt.idJudet,
 jt.denumireJudet,
 jt.idTara  -- Adăugăm idTara (referință la țara părinte)
FROM JSON_TABLE(
  (SELECT doc FROM json),
  '$.results[*].data[*].row'  -- Accesăm corect array-ul cu rezultate
  COLUMNS (
    idJudet      PATH '$[0].idjudet' NULL ON ERROR,     -- idJudet este primul element din array-ul "row"
    denumireJudet PATH '$[0].denumirejudet' NULL ON ERROR,  -- denumireJudet este al doilea element din array-ul "row"
    idTara       PATH '$[1].idtara' NULL ON ERROR      -- idTara este al treilea element din array-ul "row"
  )
) jt;
```

```
SELECT * FROM judete_view_neo4j;

-------------------------------------------------------------------------------

CREATE OR REPLACE VIEW tari_view_neo4j AS

WITH json AS (

   SELECT query_neo4j_rest_graph_data(

      'http://localhost:7474/db/neo4j/tx/commit',

      'MATCH (tara:Tara) RETURN *',

      'neo4j', 'administrator') doc

   FROM dual

)

SELECT

 jt.idTara,

 jt.denumireTara

FROM JSON_TABLE(

   (SELECT doc FROM json),

   '$.results[*].data[*].row[*]'  -- Accesăm corect array-ul cu rezultate

   COLUMNS (

      idTara         PATH '$[0].idtara' NULL ON ERROR,  -- idOras este primul element din array-ul "row"

      denumireTara   PATH '$[0].denumiretara' NULL ON ERROR   -- denumireOras este al doilea element din array-ul "row"

   )

) jt;

select * from tari_view_neo4j;


SELECT

   o.denumireOras,

   j.denumireJudet,

   t.denumireTara

FROM oras_view_neo4j o

JOIN judete_view_neo4j j ON o.idJudet = j.idJudet
```

```
JOIN tari_view_neo4j t ON j.idTara = t.idTara;


--------------------------------------------------------------------------------
--- 23 Data Source Model: CSV Views
--------------------------------------------------------------------------------



DROP DIRECTORY ext_file_ds;
CREATE OR REPLACE DIRECTORY ext_file_ds AS
'C:\Users\Administrator\Desktop\scripturi creare tabele_V1\1 Data source';
GRANT ALL ON DIRECTORY ext_file_ds TO PUBLIC;


SELECT * FROM all_directories;


DROP TABLE presonal_biblioteca;
CREATE TABLE presonal_biblioteca (
  id_angajat      NUMBER(4),
  id_biblioteca      NUMBER(4),
  nume_angajat            VARCHAR(100),
  prenume_angajat          VARCHAR(100),
  functie          VARCHAR(40),
  email_angajat     VARCHAR(100)
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY ext_file_ds
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE SKIP 1
    FIELDS TERMINATED BY ','
    MISSING FIELD VALUES ARE NULL
  )
```

```
  LOCATION ('personal_biblioteca.csv')
)
REJECT LIMIT UNLIMITED;

SELECT * FROM presonal_biblioteca;
```

CSV LINK VIEW

```
--- 23 Data Source Model: CSV Views
DROP DIRECTORY ext_file_ds;
CREATE OR REPLACE DIRECTORY ext_file_ds AS
'C:\Users\Administrator\Desktop\scripturi creare tabele_V1\1 Data source';
GRANT ALL ON DIRECTORY ext_file_ds TO PUBLIC;

SELECT * FROM all_directories;

DROP TABLE presonal_biblioteca;
CREATE TABLE presonal_biblioteca (
  id_angajat      NUMBER(4),
  id_biblioteca      NUMBER(4),
  nume_angajat            VARCHAR(100),
  prenume_angajat          VARCHAR(100),
  functie          VARCHAR(40),
  email_angajat     VARCHAR(100)
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY ext_file_ds
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE SKIP 1
    FIELDS TERMINATED BY ','
```

```
    MISSING FIELD VALUES ARE NULL
  )
  LOCATION ('personal_biblioteca.csv')
)
REJECT LIMIT UNLIMITED;

SELECT * FROM presonal_biblioteca;
```