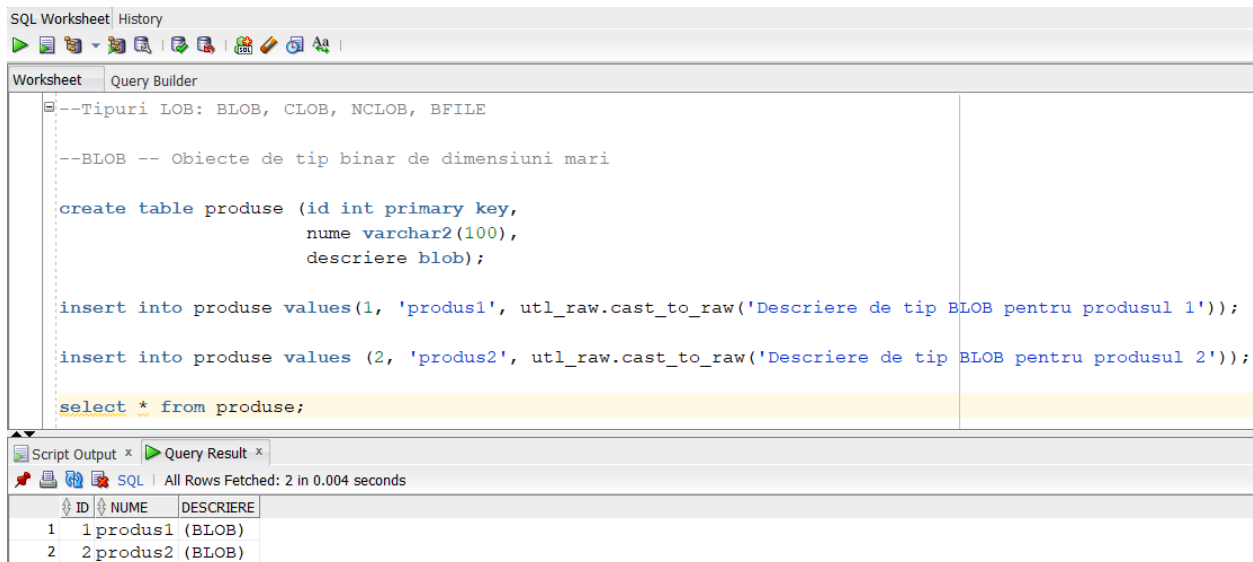


Tipuri de date LOB

- **Tipul BLOB**

Pentru a exemplifica tipul de date BLOB, creăm un tabel numit PRODUSE, cu attributele id, nume și descriere. Astfel, descrierea va fi de tip BLOB. Inserăm 2 înregistrări în tabel: produs1 și produs2.



The screenshot shows an SQL Worksheet interface with a 'Query Builder' tab. The SQL script in the editor is as follows:

```
--Tipuri LOB: BLOB, CLOB, NCLOB, BFILE  
  
--BLOB -- Obiecte de tip binar de dimensiuni mari  
  
create table produse (id int primary key,  
                     nume varchar2(100),  
                     descriere blob);  
  
insert into produse values(1, 'produs1', utl_raw.cast_to_raw('Descriere de tip BLOB pentru produsul 1'));  
  
insert into produse values (2, 'produs2', utl_raw.cast_to_raw('Descriere de tip BLOB pentru produsul 2'));  
  
select * from produse;
```

Below the script, the 'Query Result' tab shows the execution output:

Script Output x Query Result x
All Rows Fetched: 2 in 0.004 seconds

ID	NUME	DESCRIERE
1	1produs1	(BLOB)
2	2produs2	(BLOB)

Remarcăm faptul că, pentru a putea insera date de tip BLOB, avem nevoie de pachetul UTL_RAW (pachet cu funcții ce manipulează datele de tip RAW, BLOB-ul fiind, și el, de acest tip). Funcția CAST_TO_RAW este utilizată pentru a converti valorile la tipul de date RAW. De asemenea, dacă această funcție nu era folosită, primeam o eroare:

```
Error starting at line : 9 in command -  
insert into produse values(1, 'produs1', 'Descriere de tip BLOB')  
Error report -  
ORA-01465: invalid hex number
```

BLOB UPDATE

În următoarele imagini este exemplificat update-ul liniei din tabel ce are tipul de date BLOB. De asemenea, se poate observa și lungimea fiecărei descrieri: cea pentru produsul 2 a fost golită cu funcția `EMPTY_BLOB()` și are lungimea 0, iar cea pentru produsul 1 are lungimea 21.

```
--BLOB Update  
--EMPTY-BLOB() = functie ce lasa variabila de tip BLOB goala, dar initializata  
update produse  
set descriere = empty_blob()  
where id = 2;
```

Script Output x Query Result x
SQL | All Rows Fetched: 2 in 0.004 seconds

ID	NUME	DESCRIERE
1	1produs1	(BLOB)
2	2produs2	(BLOB)

View Value

Information

☒ Saved Data [External Editor](#) [Download](#) View As: ☐ Image »

Length 0

Empty false

Temporary false

Open false

☐ Local Data [Load](#) [Set NULL](#) View As: ☐ Image ☐ Text

File Name

File Size

Help OK Cancel

```
--BLOB Update  
--EMPTY-BLOB() = functie ce lasa variabila de tip BLOB goala, dar initializata  
update produse  
set descriere = empty_blob()  
where id = 2;
```

Script Output x Query Result x
SQL | All Rows Fetched: 2 in 0.004 seconds

ID	NUME	DESCRIERE
1	1produs1	(BLOB)
2	2produs2	(BLOB)

View Value

Information

☒ Saved Data [External Editor](#) [Download](#) View As: ☐ Image »

Length 21

Empty false

Temporary false

Open false

☐ Local Data [Load](#) [Set NULL](#) View As: ☐ Image ☐ Text

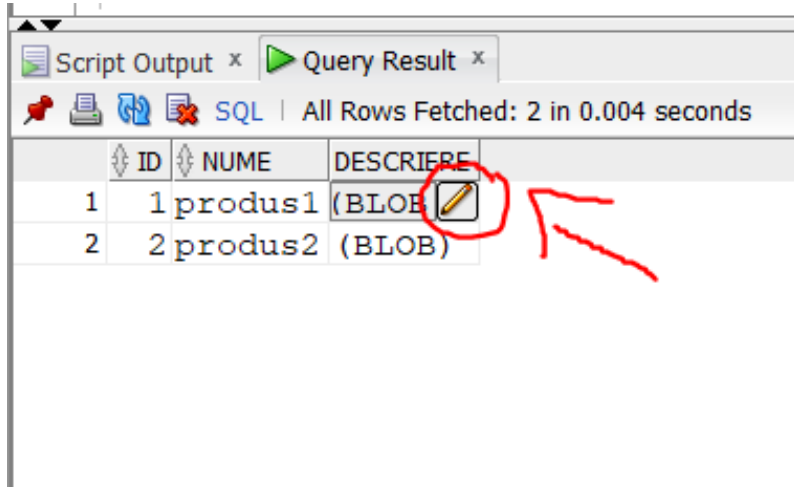
File Name

File Size

Help OK Cancel

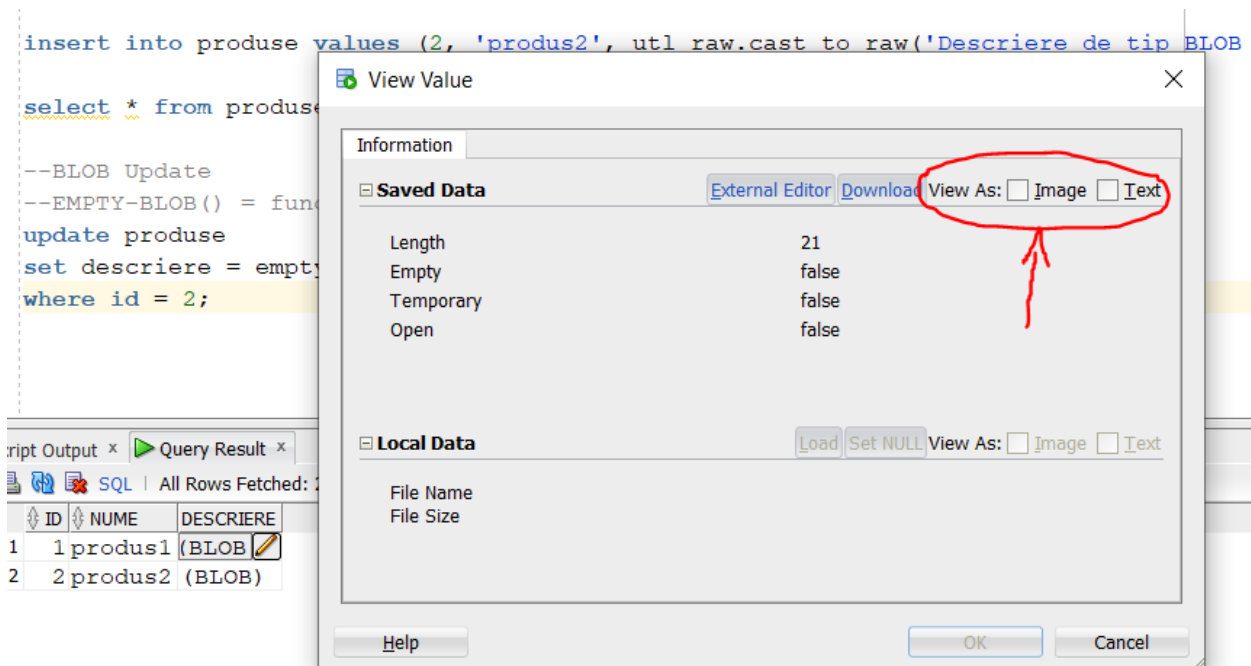
Cum putem vizualiza datele de tip BLOB?

Pasul 1:



ID	NUME	DESCRIERE
1	produs1	(BLOB)
2	produs2	(BLOB)

Pasul 2:



```
insert into produse values (2, 'produs2', utl raw.cast to raw('Descriere de tip BLOB'))
select * from produse
--BLOB Update
--EMPTY-BLOB() = fun
update produse
set descriere = empty
where id = 2;
```

ID	NUME	DESCRIERE
1	produs1	(BLOB)
2	produs2	(BLOB)

View Value

Information

Saved Data External Editor Download View As: ☐ Image ☐ Text

Length	21
Empty	false
Temporary	false
Open	false

Local Data Load Set NULL View As: ☐ Image ☐ Text

File Name	
File Size	

Help OK Cancel

Pasul 3:

```
insert into produse values (2, 'produs2', utl raw.cast to raw('Descriere de tip BLOB pe  
select * from produse  
--BLOB Update  
--EMPTY-BLOB() = fun  
update produse  
set descriere = empty  
where id = 2;
```

View Value

Information Saved Text

Descriere de tip BLOB

✓

Help OK Cancel

ID	NUME	DESCRIERE
1	1produs1	(BLOB)
2	2produs2	(BLOB)

- **TIPUL CLOB**

Pentru a exemplifica tipul de date CLOB, creăm un tabel numit JOBS, cu attributele id, nume și descriere. Astfel, descrierea va fi de tip CLOB. Inserăm 2 înregistrări în tabel: job1 și job2.

SQL Worksheet History

Worksheet Query Builder

```
--CLOB -- Obiecte de tip caracter de dimensiuni mari

create table jobs (id int primary key,
                  nume varchar2(100),
                  descriere clob);

insert into jobs values(1, 'job1', 'Descriere de tip CLOB pentru jobul 1');

insert into jobs values (2, 'job2', 'Descriere de tip CLOB pentru jobul 2');

select * from jobs;
```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.011 seconds

ID	NUME	DESCRIERE
1	1job1	Descriere de tip CLOB pentru jobul 1
2	2job2	Descriere de tip CLOB pentru jobul 2

CLOB UPDATE

```
select * from jobs;

--CLOB Update
--EMPTY-CLOB() = functie ce lasa variabila de tip BLOB goala, dar initializata

update jobs
set descriere = empty_clob()
where id = 2;
```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.005 seconds

ID	NUME	DESCRIERE
1	1job1	Descriere de tip CLOB pentru jobul 1
2	2job2	

- **Tipul NCLOB**

Pentru a exemplifica tipul de date NCLOB, creăm un tabel numit OBIECTE, cu attributele id, nume și descriere. Astfel, descrierea va fi de tip NCLOB. Inserăm 2 înregistrări în tabel: obiect1 și obiect2.

```
--NCLOB -- Obiecte de tip caracter de dimensiuni mari
-- Datele stocate corespund setului national de caractere

create table obiecte (id int primary key,
                    nume varchar2(100),
                    descriere nclob);

insert into obiecte values(1, 'obiect1', 'Descriere de tip NCLOB pentru obiectul 1');

insert into obiecte values (2, 'obiect2', 'Descriere de tip NCLOB pentru obiectul 2');

select * from obiecte;
```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.014 seconds

ID	NUME	DESCRIERE
1	obiect1	Descriere de tip NCLOB pentru obiectul 1
2	obiect2	Descriere de tip NCLOB pentru obiectul 2

NCLOB UPDATE

```
select * from obiecte;
```

```
--NCLOB Update

update obiecte
set descriere = 'Update pentru obiectul 2'
where id = 2;
```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.005 seconds

ID	NUME	DESCRIERE
1	obiect1	Descriere de tip NCLOB pentru obiectul 1
2	obiect2	Update pentru obiectul 2

Am observat anterior faptul că, la tipurile BLOB ȘI CLOB, exista o funcție de EMPTY (EMPTY_BLOB(), EMPTY_CLOB()). Tipul NCLOB nu are această funcție specifică.

- **Tipul BFILE**

Pentru a exemplifica tipul de date BFILE, creăm un tabel numit MELODII, cu attributele id, nume și link. Astfel, link-ul va fi de tip BFILE. Inserăm o înregistrare în tabel: melodie.

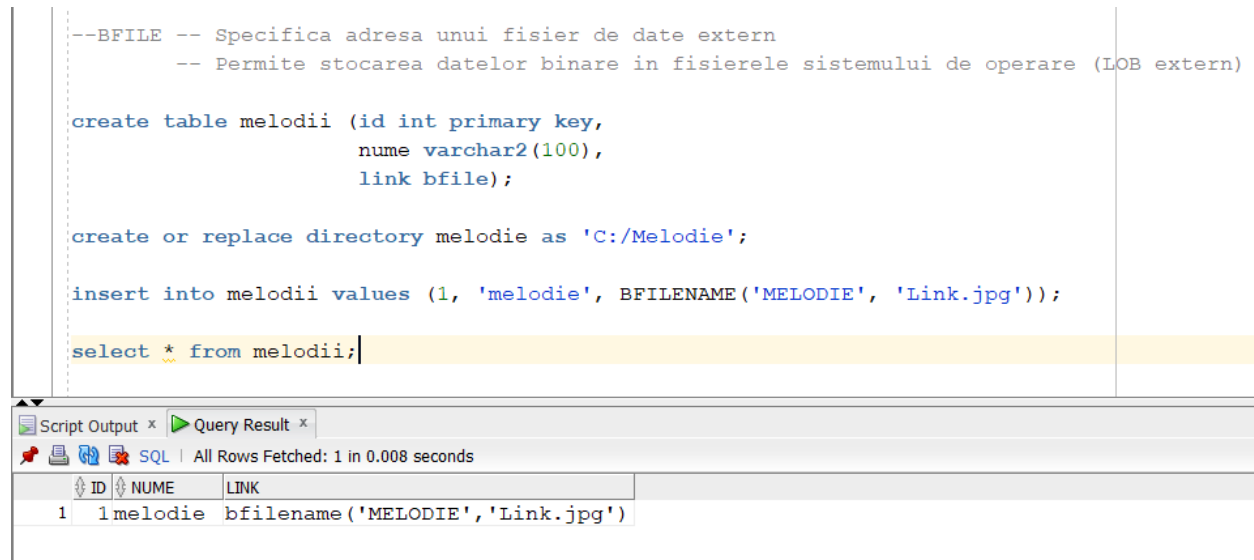
```
--BFILE -- Specifica adresa unui fisier de date extern
-- Permite stocarea datelor binare in fisierele sistemului de operare (LOB extern)

create table melodii (id int primary key,
                     nume varchar2(100),
                     link bfile);

create or replace directory melodie as 'C:/Melodie';

insert into melodii values (1, 'melodie', BFILENAME('MELODIE', 'Link.jpg'));

select * from melodii;
```



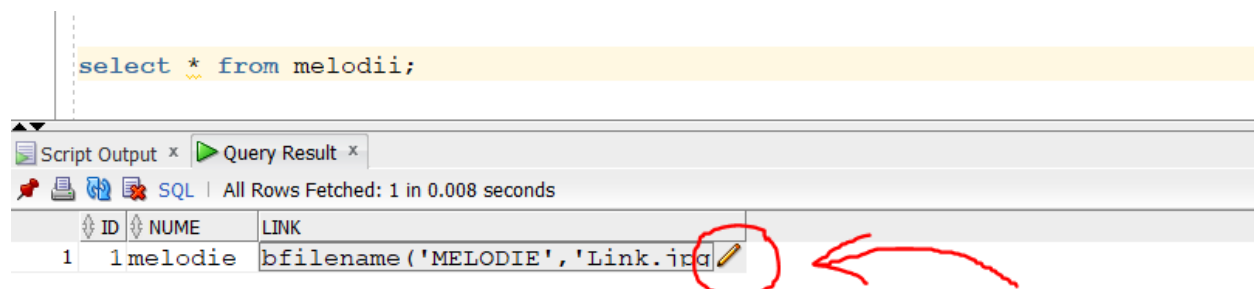
ID	NUME	LINK
1	1melodie	bfilename('MELODIE', 'Link.jpg')

Remarcăm faptul că, pentru a putea folosi tipul BFILE, avem nevoie să creăm un director. În exemplul de mai sus, directorul este MELODIE. Astfel, având directorul creat, putem insera trimeri către link-urile melodiilor în coloana LINK, dar doar cu ajutorul funcției BFILENAME(), care are ca parametri directorul și numele fișierului dorit.

Pentru a putea vizualiza dacă memorarea în BFILE s-a făcut corespunzător, procedăm astfel:

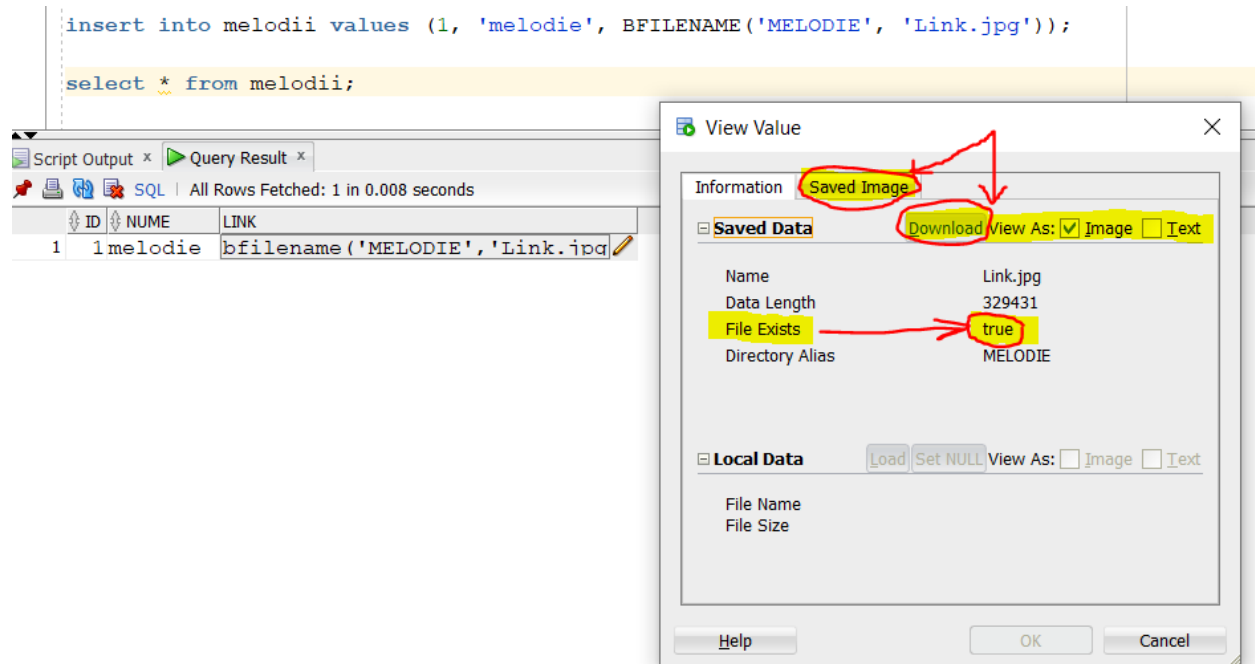
Pasul 1:

```
select * from melodii;
```



ID	NUME	LINK
1	1melodie	bfilename('MELODIE', 'Link.jpg')

Pasul 2:



De asemenea, după cum se observă în imaginea anterioară, fișierul poate fi și descărcat, dar și vizualizat direct în program.

Așadar, tipurile de date LOB sunt foarte folosite pentru stocarea unor date de dimensiuni foarte mari, precum caractere, fișiere și multe altele.