

1) Constrângeri

Constrângerile sunt niște reguli definite prin care putem crește gradul de corectitudine și încredere asupra datelor din baza de date [FO]. Sarcina respectării constrângerilor revine SGBD-ului.

În capitolul precedent am văzut cum funcționează constrângerea de cheie primară (PRIMARY KEY) a unei tabel, și ce reprezintă aceasta (conformitatea cu modelul relațional, în care tabelele sunt mulțimi de linii, prin consecință fiecare linie este un unicat). Cheia primară identifică în mod unic (prin valoarea ei unică) o linie a tabelului.

Cheia primară este aleasă din mai multe *chei candidate* posibile. De obicei se alege cheia care are lungimea minimă (numărul minim de atribute). O cheie candidată care nu a fost aleasă cheie primară se numește *cheie secundară* sau *cheie alternativă*. O *super-cheie* este o mulțime de coloane care îndeplinește condiția de cheie (unicitate) și care conține cel puțin o submulțime care este cheie.

Cheia secundară

Exercițiu: creați o tabelă Cărți cu următoarea comandă:

```
CREATE TABLE Carti (  
    Cod          NUMBER(4) PRIMARY KEY,  
    Titlu        VARCHAR(20),  
    Autor        VARCHAR(20),  
    Data_ap      DATE CHECK (Data_ap > DATE'2000-01-01'),  
    UNIQUE (Titlu, Autor, Data_ap) );
```

Clauza „UNIQUE” folosită la crearea tabeli Cărți va seta o cheie secundară formată din attributele (Titlu, Autor, Data_ap).

Constrângerea CHECK (*check constraint*)

Observați că pentru atributul Data_ap a fost setată o *constrângere de tip verificare* (CHECK). SGBD-ul va permite introducerea de linii doar în cazul în care condiția CHECK are valoarea TRUE.

Exercițiu:

- Inserați câteva linii în tabela Cărți. Încercați să inserați linii cu aceleași valori pentru Titlu, Autor și Data apariției. De ce nu se poate?
- Inserați câteva linii nespecificând valori pentru Titlu, Autor și/sau Data_ap. Observați diferența față de cheia primară, care nu permite valori NULL.
- Verificați funcționarea constrângerii CHECK pentru coloana Data_ap.

Cheia străină

O cheie străină este o constrângere inter-relații (între mai multe tabele) și constă într-unul sau mai multe atribute dintr-o relație R₁ (cea care conține cheia străină) care referă unul sau mai multe atribute cu aceeași semnificație și tip de domeniu (tip de date în Oracle) dintr-o relație R₂ (relația referită). Pentru simplitate *se preferă folosirea aceluiași nume* pentru atributul (coloana) / attributele din R₁ care reprezintă cheia străină și pentru coloana / coloanele din R₂ care reprezintă cheia candidată referită.

Testați direct în SQL Developer următorul exemplu de legătură între 2 tabele printr-o cheie străină:

```
CREATE TABLE Clienti (
    Cod_client NUMBER(4) PRIMARY KEY,
    Nume VARCHAR2(20) NOT NULL,
    Adresa VARCHAR2(20),
    Telefon VARCHAR2(15));

CREATE TABLE Facturi (
    Nr_fact NUMBER(4) PRIMARY KEY,
    Data_fact DATE,
    Cod_client NUMBER(4),
    Valoare NUMBER(5),
    FOREIGN KEY (Cod_client) REFERENCES Clienti (Cod_client));
```

Tabela Clienti, în care coloana Cod_client este cheie primară, se numește *tabelă părinte* (relația R₂ de mai sus, sau relația referită).

Tabela care conține cheia străină (foreign key), în cazul nostru Facturi (R₁ – cea care referă), se numește *tabelă copil*.

În Fig. 1 se poate vedea structura tabelor și asocierea Facturi – Clienti care este de tip N:1 (un client poate avea mai multe facturi, dar pe o factură poate fi trecut un singur client).

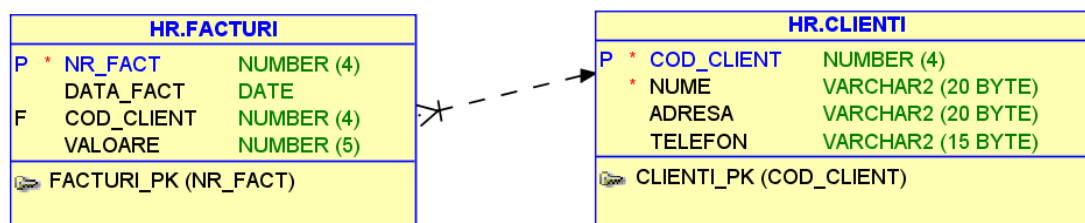


Fig. 1. Tabela părinte “Clienti” și tabela copil “Facturi”

Alternativ, putem denumi constrângerea de cheie străină (de exemplu cu numele FK_Facturi, dar se poate stabili orice nume), la crearea tablei, după cum urmează:

```
CREATE TABLE Facturi (
    Nr_fact NUMBER(4) PRIMARY KEY,
    Data_fact DATE,
    Cod_client NUMBER(4),
    Valoare NUMBER(5),
    CONSTRAINT FK_Facturi FOREIGN KEY (Cod_client) REFERENCES Clienti (Cod_client));
```

Constrângerea se poate adăuga și ulterior (deși în principiu se recomandă stabilirea tuturor constrângerilor la crearea tabelor), prin modificarea structurii tablei:

```
ALTER TABLE Facturi ADD CONSTRAINT FK_Facturi
    FOREIGN KEY (Cod_client) REFERENCES Clienti (Cod_client);
```

Exercițiu: introduceți câteva linii, întâi în tabela părinte, apoi în tabela fiu. De exemplu tabelele completate ar putea avea liniile ca în Fig. 2.

FACTURI				CLIENTI			
Nr_fact	Data_fact	Cod_client	Valoare	Cod_client	Nume	Adresa	Telefon
10	14-OCT-12	3	500	1	Ionescu Andrei	Strada Florilor	1234
14	18-OCT-12	1	1500	2	SERBAN Daniel	Strada Lunga	1247
17	18-OCT-12	3	1000	3	Popescu Tudor	Str. Siret	4510
20	19-OCT-12	3	1040	4	VLAD Daniel	Str. Sibiu	2341
22	20-OCT-12	4	1300				

Fig. 2. Exemplu de stare a tabelor Clienți și Facturi

Exercițiu:

- Încercați să introduceți în tabela Facturi o linie cu o valoare pentru Cod_client = 5.
- Încercați să ștergeți din tabela Clienți linia cu Cod_client = 1.

De ce nu este posibil?

- Încercați să introduceți în tabela Clienți o linie cu o valoare pentru Cod_client = 5 (să adăugați un nou client).

- Încercați să ștergeți din tabela Facturi linia cu data facturii de 19-OCT-2012.

Observați că aceste două ultime operații sunt posibile.

Explicație: cheia străină definită creează o *constrângere referențială* (*referential integrity constraint*). În tabela Facturi nu se poate introduce un cod client inexistent în tabela Clienți cu o singură excepție: valoarea NULL. Iar din tabela clienți nu se poate șterge / modifica o linie care are un cod client existent în tabela Facturi. Bineînțeles, nici instrucțiunea *drop table clienti*; nu va avea efect, fiind interzisă în acest caz de SGBD. Acest efect poate fi schimbat prin activarea opțiunii „CASCADE”.

Stabilirea modului de ștergere sau actualizare a liniilor / valorii cheii referite din tabela părinte se poate face prin trei opțiuni: RESTRICT (setare implicită), „CASCADE” sau „SET NULL”.

Setarea unei opțiuni CASCADE va antrena, la ștergerea liniei cu o anumită valoare pentru cod_client din tabela Clienți, ștergerea tuturor liniilor cu aceeași valoare pentru cod_client din tabela Facturi. În mod similar, o actualizare a cod_client în Clienți va determina actualizarea cheii străine cod_client în liniile corespunzătoare din tabela Facturi.

Opțiunea CASCADE se stabilește la crearea tabelului, după cum urmează:

```
CREATE TABLE Facturi (
    Nr_fact    NUMBER(4)    PRIMARY KEY,
    Data_fact  DATE,
    Cod_client NUMBER(4),
    Valoare    NUMBER(5),
    FOREIGN KEY (Cod_client) REFERENCES Clienți (Cod_client) ON DELETE CASCADE );
```

Pentru a stabili o actualizare cu opțiunea „cascade”, se poate specifica și: „ON UPDATE CASCADE” (totuși, Oracle nu permite opțiunea propriu zisă, ci un mecanism alternativ).

Opțiunea „SET NULL”, folosită în loc de „CASCADE”, în loc de a șterge liniile din tabela fiu, va seta valoarea cheii străine la NULL.

Dacă am uitat să stabilim opțiunile de tip „cascade” sau „set null” la crearea tabelului, putem să le adăugăm ulterior (chiar dacă avem linii inserate în tabelă) folosind următoarea procedură:

- ștergem cheia străină prin comanda

```
ALTER TABLE Facturi DROP CONSTRAINT FK_Facturi;
```

- adăugăm constrângerea de cheie străină cu toate opțiunile dorite, de exemplu:

```
ALTER TABLE Facturi ADD CONSTRAINT FK_Facturi
```

```
FOREIGN KEY (Cod_client) REFERENCES Clienti (Cod_client) ON DELETE CASCADE;
```

Această metodă funcționează doar dacă am denumit constrângerea de cheie străină (în exemplele de mai sus am folosit numele FK_Facturi).

Exercițiu: testați funcționarea opțiunii „on delete cascade” sau „on delete set null”.

2) Proiectarea conceptuală a bazelor de date

Pentru a dezvolta cu succes o bază de date, trebuie parcurse câteva etape de proiectare, cum ar fi:

- analiza cerințelor (colectarea informațiilor privind datele stocate, funcționalitățile posibile ale sistemului, utilizarea informațiilor);
- proiectarea conceptuală a bazei de date;
- proiectarea logică a bazei de date (alegerea unui SGBD și transcrierea modelului conceptual de nivel înalt în modelul relațional DDL oferit de SGBD);
- etape avansate, precum: rafinarea modelului (îmbunătățiri), creșterea performanțelor de interogare (de exemplu prin adăugarea de indexuri), creșterea securității etc.

Proiectarea conceptuală a unei baze de date presupune proiectarea schemei conceptuale și a schemelor externe.

Modelul Entitate-Asociere (E-A) (sau Entitate-Relație) este modelul conceptual de nivel înalt, independent de sistemul de gestiune, larg răspândit în proiectarea schemei conceptuale a unei baze de date. Modelul E-A descrie entitățile puternice sau slabe (dependente) din baza de date și relațiile (asocierile) posibile între aceste entități. Diagrama E-A este realizată după o analiză amănunțită a cerințelor bazei de date.

Fiecare entitate modelează un aspect al lumii reale (o categorie de persoane, activități, obiecte etc.), de exemplu entitatea „Student”, și este reprezentată printr-un substantiv.

O asociere este o corespondență între două (asocieri binare) sau mai multe entități și e reprezentată printr-un verb, de exemplu: „studiază la”.

Asocierile binare pot fi: unu-la-unu (1:1), unu-la-multe (1:N) sau multe-la-multe (M:N).

Entitățile (dar uneori și asocierile) sunt caracterizate (descrise) prin mai multe atribute. De exemplu un student poate avea următoarele atribute: CNP, nume, prenume, data nașterii etc.

Din punct de vedere grafic, în diagrama E-A entitățile sunt încadrate într-un dreptunghi, iar asocierile sunt reprezentate prin romburi. Atributele sunt reprezentate prin elipse ce încadrează numele fiecărui atribut, conectate prin linii la entitățile respective.

Problemă

Fie diagrama (schema conceptuală) Entitate-Asociere din Fig. 1, care reprezintă un model simplificat al gestiunii consultațiilor într-o policlinică.

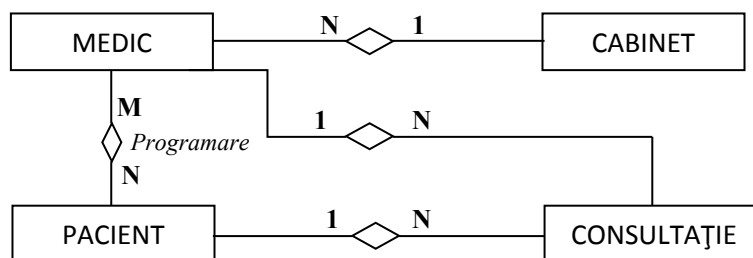


Fig. 1 Diagrama Entitate - Asociere

Diagrama de mai sus definește patru entități (Medic, Pacient, Cabinet și Consultație) și patru asocieri binare între entități.

Observație: trei asocieri sunt de tip 1:N, iar una este de tip M:N cu numele „Programare”.

Asocierile 1:N pot fi reprezentate prin chei străine (foreign key sau FK).

Motivarea alegerii asocierilor:

- La un cabinet pot consulta mai mulți medici (în zile diferite, ture diferite), dar un medic nu consultă decât la un singur cabinet (1:N);
- Un pacient poate fi consultat de mai multe ori, dar în cadrul unei consultații nu se consultă decât un singur pacient (1:N);
- Un medic poate avea mai mulți pacienți programați, iar un pacient poate face programări la mai mulți medici (M:N);
- Un pacient poate avea mai multe consultații, dar o consultație se face cu un singur pacient.

Atenție: Pentru a respecta modelul relațional și a nu avea linii duplicate (cel puțin din punct de vedere semantic) în tabele (chiar dacă de exemplu pe o linie vom avea același medic și atributele vor diferi doar prin valoarea unui atribut, de exemplu: cod_pacient), **se recomandă modelarea unei asocieri M:N printr-o nouă tabelă.**

Astfel asocierea denumită „Programare” în diagrama E-A din Fig. 1 va deveni tabela Programare, o tabelă de legătură între entitățile Medic și Pacient.

Proiectarea relațiilor și a atributelor lor:

1) CABINET (cod_cabinet, nume, dotări)

Cheia primară este cod_cabinet.

2) MEDIC (cod_medic, nume, prenume, specializare, cod_cabinet)

Cheia primară este cod_medic. Cod_cabinet este cheie străină care va face legătura cu tabela Cabinet.

3) PACIENT (cod_pacient, cnp, nume, prenume, tel)

Cod_pacient este cheie primară artificială, dar am fi putut alege și CNP-ul drept cheie primară naturală.

4) CONSULTAȚIE (cod_consultatie, cod_pacient, cod_medic, data_consult, mentiuni)

Cod_consultatie este cheie primară. Cod_pacient și cod_medic sunt chei străine.

5) PROGRAMARE (cod_pacient, cod_medic, data_programarii)

Cod_pacient și cod_medic sunt chei străine. Cheia primară poate fi în acest caz doar compusă (din cele 3 atribute).

Faza de proiectare se finalizează cu realizarea schemei logice care poate arăta de exemplu precum cea din Fig. 2 (schemă realizată în SQL Developer).

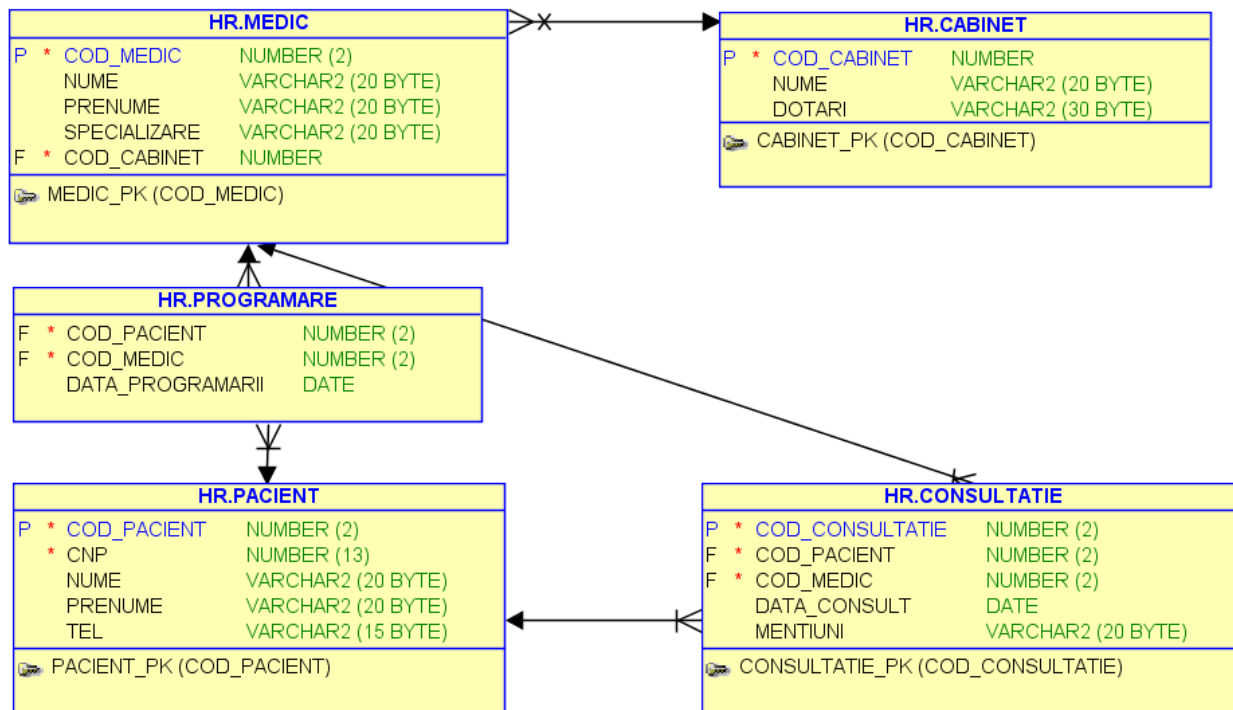


Fig. 2 Schema logică aferentă diagramei E-A din Fig. 1

Observație: asocierile binare în SQL Developer sunt reprezentate cu 3 liniuțe în partea multiplicității (cardinalității) N sau M, și cu o singură linie în partea cardinalității 1.

Exercițiu:

- Scrieți în SQL Developer și rulați instrucțiunile CREATE TABLE pentru a crea tabelele din schema din Fig. 2. Ordinea de crearea a tabelelor trebuie să fie: cabinet, medic, pacient, consultație și programare. Stabiliți cheile necesare și setați opțiunea „SET NULL” sau „CASCADE” pentru operațiile de ștergere și/sau actualizare. Puteți seta și alte constrângeri, de exemplu NOT NULL sau CHECK.
- Introduceți mai multe linii în toate tabelele create. Un exemplu de stare a celor 5 tabele poate fi cel din Fig. 3.

Cabinet

Cod_cabinet	Nume	Dotari
1	Cardiologie	EKG, Echograf
2	Oftalmologie	Oftalmoscop, Tonometru
3	Medicina Interna	

Programare

Cod_pacient	Cod_medice	Data_programarii
1	1	23-OCT-12
3	2	20-OCT-12

Medic

Cod_medice	Nume	Prenume	Specializare	Cod_cabinet
1	Popescu	Florin	Cardiolog	1
2	Ionescu	Dan	Cardiolog	1
3	Ionescu	Mihnea	Oftalmologie	2

Pacient				
Cod_pacient	CNP	Nume	Prenume	Tel
1	12345	Popescu	Vasile	0124
2	17144	Manolescu	Dan	0425
3	27144	Florescu	Adina	0745

Consultație				
Cod_consultatie	Cod_pacient	Cod_medic	Data_consult	Mentiiuni
1	1	1	23-OCT-12	observatie 1
2	3	2	20-OCT-12	observatie 2

Fig.3. Exemplu de stare a celor 5 tabele

- Încercați să modificați, apoi să ștergeți codul unui cabinet, al unui pacient sau al unui medic și observați modul de funcționare al constrângerilor referențiale (afișați conținutul tabelelor care sunt afectate de modificări).

Notă: interesant este de văzut efectul constrângerilor cu setarea CASCADE sau SET NULL.

3) Câteva comenzi ajutătoare

Comanda SQL cu ajutorul căreia se poate afișa structura unei tabele este DESCRIBE.

Exemplu de utilizare:

Describe clienti;

Va avea ca efect afișarea:

```


Name      Null    Type
-----
COD_CLIENT NOT NULL NUMBER(5)
NUME       VARCHAR2(20)
ADRESA     VARCHAR2(20)
TELEFON    CHAR(14)

```

Adică va afișa un tabel cu trei coloane: numele coloanelor, dacă sunt permise valori NULL și tipul de date. Coloanele vor fi afișate exact în ordinea definirii tablei (cu create table). Astfel, comanda este utilă când dorim să inserăm linii nespecificând numele coloanelor, ci doar precizând toate valorile, în ordinea în care au fost definite coloanele (de exemplu: cod_client, nume, adresa, telefon).

Dacă dorim să aflăm / verificăm care sunt cheile primare / candidate / străine asociate unei anumite tabele, putem da dublu-click pe numele tablei (de exemplu Facturi) în panoul din stânga al SQL Developer, apoi alege tab-ul "Constraints", cum este prezentat în Fig. 4. În coloana "R_TABLE_NAME" va fi afișat numele tablei părinte, în cazul nostru "Clienti", iar cu un click pe această valoare, se va afișa mai jos coloana ce reprezintă cheia primară referită (în cazul nostru cod_client).

Tab-ul "Data" va afișa conținutul tablei respective (echivalent cu select * from facturi), dar, atenție, spre deosebire de rularea comenzii SQL, este posibil ca datele să nu fie actualizate (să nu reflecte starea tablei reale stocate în baza de date).

Afișarea a mai multor tabele în același ecran se poate face apăsând pe butonul "Freeze view" () înainte de a da dublu-click pe numele unei tabele din panoul din stânga.

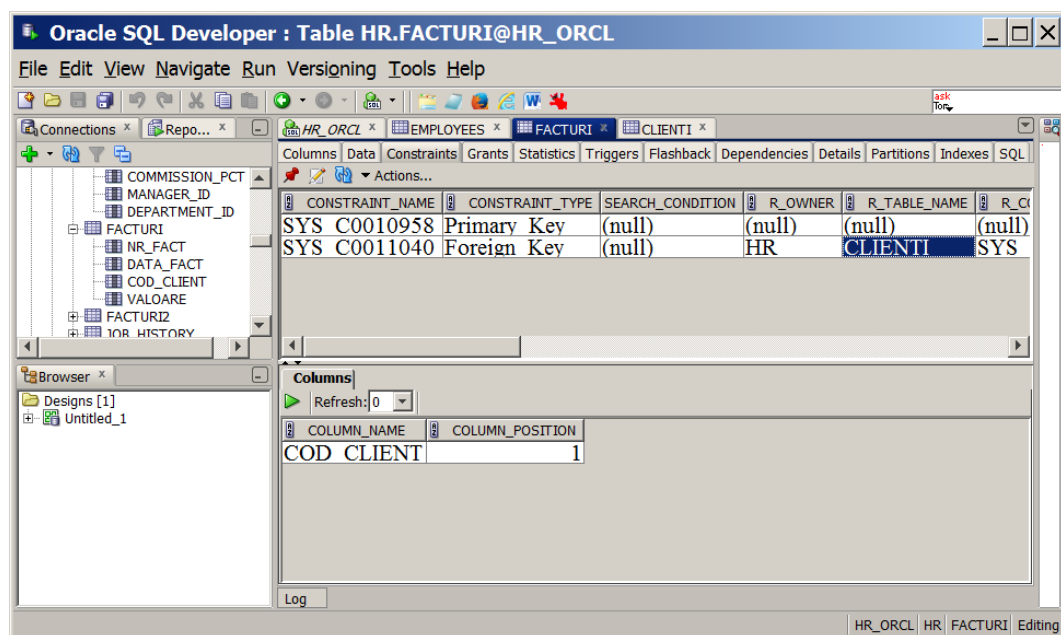



Fig.4. Exemplu de afișare a constrângerilor pentru tabela Facturi

Pentru a șterge toate liniile de text din fereastra de editare de comenzi, sau din fereastra "Script Output" (inclusiv mesaje de eroare) putem folosi butonul .

4) Utilizarea „Data Modeler” din SQL Developer

a) Generarea automată a schemelor logice pornind de la tabele existente

În SQL Developer putem crea scheme logice (precum cea din Fig. 2) pornind de la tabelele deja existente, după cum urmează:

- Din meniul principal alegeți: View - Data Modeler – Browser.
Se va deschide o fereastră în partea stânga – jos cu titlul Browser
- Din lista de elemente afișată, cu click dreapta pe "Relational Models" se alege "New relational model"
- Se selectează numele noului model relațional și din meniul ce apare prin click dreapta, se alege Show.
Se va deschide fereastra pe care va fi afișată schema logică
- Din panoul din stânga (Connections) se face drag-and-drop cu mouse-ul pentru tabelele pe care dorim să le plasăm pe schemă. De exemplu Secții și Angajați
- Cu click dreapta pe numele modelului din fereastra Browser se alege "Discover Foreign Keys"

Rezultatul obținut poate fi observat în Fig. 5.

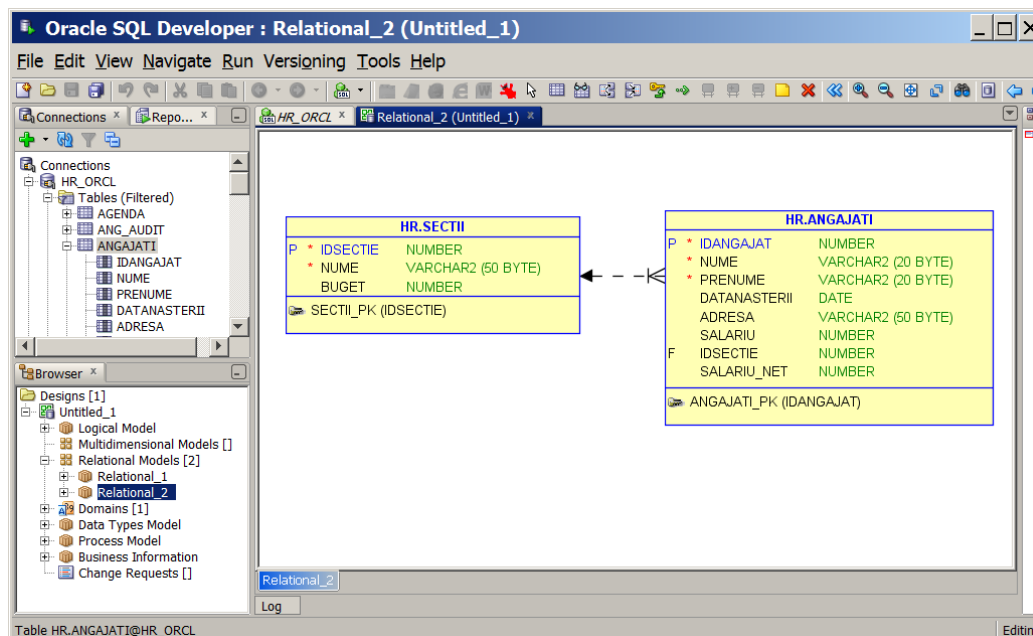


Fig.5. Exemplu de afișare a schemei logice pentru tabelele Angajați și Secții

b) Crearea automată a tabelelor pornind de la o schemă logică

SQL Developer oferă facilitatea de a genera automat codul SQL necesar creării tabelelor, pornind de la o schemă logică.

Pentru început deschidem fereastra Browser prin comanda: View - Data Modeler – Browser, ca la punctul a). Apoi alegem crearea unui nou model (unei noi scheme logice), "New relational model"

Pentru a adăuga o nouă tabelă, din bara de instrumente vom alege "New Table" ().

Va apărea o fereastră cu numele "Table Properties", ce conține mai multe rubrici (General, Columns, Primary key, Unique constraints etc.). În rubrica General putem alege numele tablei.

În Fig. 6 este prezentat un exemplu de completare a rubricii "Columns".

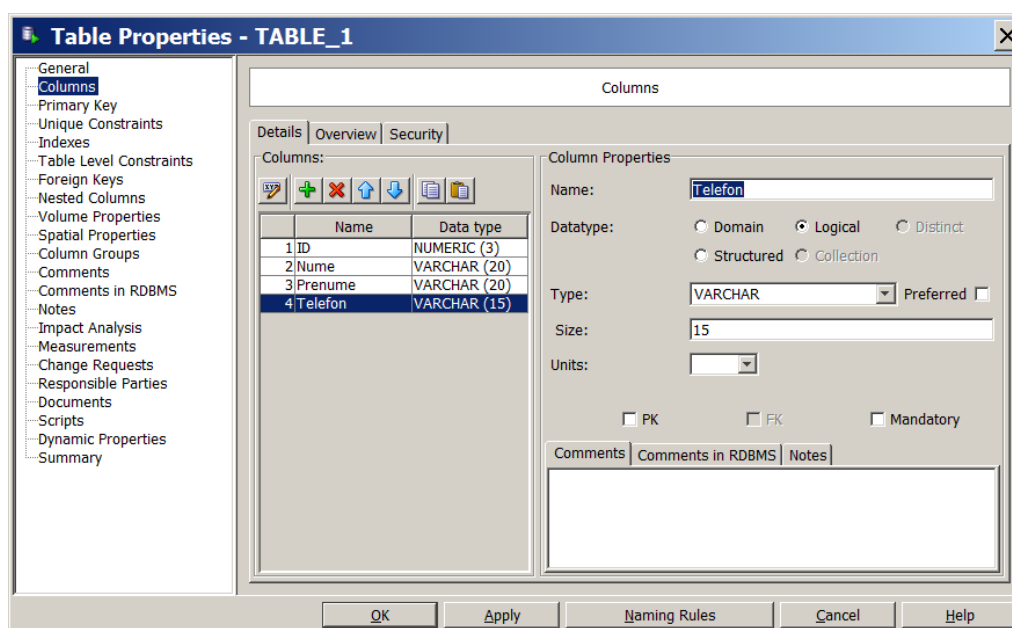


Fig.6. Stabilirea proprietăților noii tabele

Astfel, în acest exemplu, au fost definite 4 coloane: ID (cheie primară), Nume, Prenume și Telefon. Tipul de date a fost ales "Logical", iar la Type se poate specifica: Numeric, Varchar, Date etc. Pentru a seta cheia primară (ID), s-a bifat "PK" (primary key).

După finalizarea setărilor pentru tabelă, se va obține o schemă logică similară cu cea din Fig. 7.



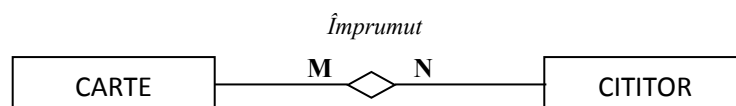
Fig.7. Diagrama tabelii Agenda_telefon

Deocamdată tabela există doar sub forma schemei logice din Fig. 7.

Pentru a genera comenzile SQL de creare a tabelii vom apăsa butonul "Generate DDL" din bara de instrumente, iar în fereastra nou deschisă se va apăsa butonul "Generate", apoi OK. În fereastra va apărea codul SQL necesar creării tabelii. Acest cod îl putem apoi copia și rula în fereastra principală a SQL Developer.

Probleme propuse

- 1) Desenați diagrama E-A pentru entitățile Jucător și Echipă, și relația dintre acestea. Se presupune că un jucător nu joacă la mai multe echipe. Stabiliți schema logică și structura celor două tabele, impunând o constrângere referențială. Un jucător poate avea de exemplu atributele: cod, nume, prenume etc. iar o echipă: cod_echipa, nume, localitate, adresa etc. Creați în SQL Developer cele 2 tabele cu comanda CREATE TABLE, setând pentru constrângerea referențială opțiunea „SET NULL”.
- 2) Introduceți câteva linii în tabelele create la punctul 1) și observați funcționarea constrângerii referențiale în cazul introducerii, actualizării și ștergerii de linii.
- 3) Fie diagrama E-A următoare



Stabiliți care sunt tabelele necesare, atributele lor, cheile primare și străine. Creați tabelele (CREATE TABLE) alegând una din constrângeri: implicită (restrict, cascade sau set null). Introduceți câteva linii în tabele pentru a simula operația de împrumut a unor cărți într-o bibliotecă.

- 4) Fie diagrama E-A următoare



Între cele 3 entități există următoarele asocieri:

- la o farmacie pot lucra mai mulți farmaciști;
- farmaciile comandă medicamente din entitatea depozit Medicamente.

Creați tabelele și constrângerile asociate diagramei de mai sus. Introduceți câteva linii.