**Universitatea “Politehnica” din Bucureşti**

**Facultatea de Electronică, Telecomunicaţii şi Tehnologia**

**Informaţiei**

**PROIECTAREA ȘI IMPLEMENTAREA UNUI SISTEM INFORMATIC PENTRU O FIRMĂ DE RECRUTARE A PERSONALULUI**

**Conducător ştiinţific:**

**Conf. dr. ing. GALAȚCHI Dan**

**Student:**

**TEODOSIADE Victor George**

**Bucureşti**

**2021**

Cuprins

[CAPITOLUL 1 4](#_Toc74084831)

[1.1 Componentele unui Sistem de baze de Date 5](#_Toc74084832)

[1.2 Arhitectura interna a unui Sistem de Baze de Date 6](#_Toc74084833)

[1.3 Avantaje oferite de Sistemele de BD 7](#_Toc74084834)

[1.4 Clasificarea Sistemelor de Baze de Date 7](#_Toc74084835)

[1.5 Modelarea Datelor 8](#_Toc74084836)

[1.6 Modelul entitate – Asociere 9](#_Toc74084837)

[1.7 Asocieri 9](#_Toc74084838)

[1.8 Categorii de asocieri binare 10](#_Toc74084839)

[1.9 Cardinalitatea asocierilor 11](#_Toc74084840)

[1.10 Diagrama Entitate-Asociere 11](#_Toc74084841)

[1.11 Modelul Entitate – Asociere Extins 12](#_Toc74084842)

[1.12 Modelul de date ierarhic 13](#_Toc74084843)

[1.13 Baze de date ierarhice 13](#_Toc74084844)

[1.14 Modelul de date retea 14](#_Toc74084845)

[1.15 Modelul de date relational 15](#_Toc74084846)

[1.16 Modelul obiect obiect-orientat 15](#_Toc74084847)

[1.17 Complexitatea datelor si a interogarilor 16](#_Toc74084848)

[1.18 Evolutia sistemelor de baze de date 16](#_Toc74084849)

[Capitolul II – BAZE DE DATE RELAȚIONALE 18](#_Toc74084850)

[2.1 Bază de Date Relaționale. Definiție. Relații. Atribute. Domenii 18](#_Toc74084851)

[2.1.1 BD – Definiție 18](#_Toc74084852)

[2.1.2 Atribute. Relații. Domenii 18](#_Toc74084853)

[2.2 Limbajul SQL 19](#_Toc74084854)

[2.2.1 Caracteristici Generale 19](#_Toc74084855)

[2.2.2 Structura Lexicală. Expresii și Operatori 20](#_Toc74084856)

[2.2.3 Funcții și Tipuri de Date SQL 21](#_Toc74084857)

[2.2.4 Domenii și Instrucțiuni SQL 22](#_Toc74084858)

[CAPITOLUL III: PROIECTAREA ȘI IMPEMENTAREA BAZEI DE DATE 24](#_Toc74084859)

[3.1 Relația entitate – atribut 24](#_Toc74084860)

[3.2 Modelul entitate - atribut 27](#_Toc74084861)

[3.3 Constrangeri 29](#_Toc74084862)

[3.4 Cheile din tabelă 29](#_Toc74084863)

[3.5 Etapa alegerii SGBD-ului 31](#_Toc74084864)

[3.6 Crearea Tabelelor 31](#_Toc74084865)

[3.6.1 Crearea și alterarea unei tabele 31](#_Toc74084866)

[3.6.2 Crearea tabelei Client 33](#_Toc74084867)

[3.6.3 Crearea tabelei Adresa 34](#_Toc74084868)

[3.6.4 Crearea tabelei Factura 35](#_Toc74084869)

[3.6.5 Crearea tabelei Contract 35](#_Toc74084870)

[3.6.6 Crearea tabelei Telefon 36](#_Toc74084871)

[3.6.7 Crearea tabelei Serviciu 36](#_Toc74084872)

[3.6.8 Crearea tabelei Prepay 37](#_Toc74084873)

[3.6.9 Crearea tabelei Abonament 37](#_Toc74084874)

[3.6.10 Crearea tabelei Subscriptie 38](#_Toc74084875)

[3.6.11 Crearea tabelei Cost 39](#_Toc74084876)

[3.7 Introducerea de constrângerilor de tip Foreign Key 40](#_Toc74084877)

[3.7.1 Introducerea cheii străine idSubsc și a cheii idAdresaîn tabela Client 40](#_Toc74084878)

[3.7.2 Introducerea cheii străine idFactura și cheii idTelefonîn tabela Contract 40](#_Toc74084879)

[3.7.3 Introducerea cheii străine idServiciu și a cheii idCostîn tabela PrePay 41](#_Toc74084880)

[3.7.4 Introducerea cheii străine idCost și a cheii idServiciu în tabela Abonament 41](#_Toc74084881)

[3.7.5 Introducerea cheii străine idTelefon în tabela Cost 41](#_Toc74084882)

[3.7.6 Introducerea cheii străine idClient, idAbonament, idPrepay, idCost, idTelefon și a cheii idContract în tabela Subscriptie 41](#_Toc74084883)

[3.8 Inspectarea tabelelor prin ”Schemă Browser” 42](#_Toc74084884)

[3.9 Inserarea datelor în tabele 47](#_Toc74084885)

[3.9.1 Introducerea datelor în tabela Telefon 48](#_Toc74084886)

[3.9.2 Introducerea datelor în tabela Factură 48](#_Toc74084887)

[3.9.3 Introducerea datelor în tabela Adresă 49](#_Toc74084888)

[3.9.4 Introducerea datelor în tabela Contract 50](#_Toc74084889)

[3.9.5 Introducerea datelor în tabela Cost 51](#_Toc74084890)

[3.9.6 Introducerea datelor în tabela Serviciu 52](#_Toc74084891)

[3.9.7 Introducerea datelor în tabela PrePay 53](#_Toc74084892)

[3.9.8 Introducerea datelor în tabela Abonament 53](#_Toc74084893)

[3.9.9 Introducerea datelor în tabela Client 54](#_Toc74084894)

[3.9.10 Introducerea datelor în tabela Subscripție 55](#_Toc74084895)

[3.10 Creare Vederi 55](#_Toc74084896)

[3.10.1 Crearea vederii de Roaming 57](#_Toc74084897)

[3.10.2 Crearea vederii cu clienții care au platit peste o anumita suma de bani 58](#_Toc74084898)

[3.10.3 Crearea vederii vechime pentru Abonați 59](#_Toc74084899)

[BIBLIOGRAFIE 60](#_Toc74084900)

# CAPITOLUL 1

**INTRODUCERE**

Bazele de date se folosesc in aproape toate domeniile de activitate actuale:

●Activitati bancare si comerciale (depozite bancare, vanzari produse)

●Productie (gestiunea stocurilor, gestiunea financiar-contabila, salarizare etc.)

●Evidenta populatiei, taxe si impozite

●Servicii (servicii medicale, rezervari bilete de calatorie etc.)

Definitie (în sens larg): O baza de date(database) este o colecţie de date corelate din punct de vedere logic, care reflectaun anumit aspect al lumii reale şi este destinataunui anumit grup de utilizatori. In acest sens pot fi considerate ca fiind “baze de date”:

●Fise de evidenta (mentinute manual)

●Fisiere de documente sau foi de calcul tabelar (Microsoft Word, Microsoft Excel)

●Baze de date mentinute computerizat

Definitie (în sens restrans): O bază de date este o colecţie de date creată şi menţinută computerizat, care permite operaţii de:

●Introducere(insert)

●Stergere(delete)

●Actualizare (update)

●Interogare (query)

## 1.1 Componentele unui Sistem de baze de Date

Un sistem de baze de date(Database System)este un sistem computerizat de menţinere a evidenţei unei anumite activităţi, folosind baze de date.

Componentele unui sistem de baze de date sunt: hardware, software, utilizatori si date persistente

􀂄Hardware:

●Sistemele de baze de date sunt instalate pe calculatoare de uz general

●Bazele de date sunt memorate fizic ca fisiere pe discuri magnetice (hard-discuri)

􀂄Software:

●Sisteme de operare, biblioteci, instrumente de dezvoltare, interfete

●Sistemul de Gestiune a Bazelor de Date(SGBD) (Database Management System –DBMS) -recepţionează cererile utilizatorilor de acces la baza de date, le interpretează, execută operaţiile corespunzătoare şi returnează rezultatul

●Aplicatii de baze de date: (Database Applications)–sunt programe care oferă anumiteutilizarialeunei baze de date

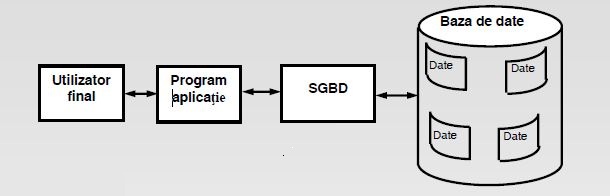


Figura 1. 1 Componentele unui sistem de baze de date

Utilizatori:

●Programatori de aplicatii

●Utilizatori finali

●Administratorul bazei de date

●Analisti si proiectanti ai bazelor de date

􀂄Datele persistente –sunt memorate in fisiere pe hard-disk

􀂄Limbaje conceptuale pentru lucrul cu bazele de date:

●Limbaje pentru Definirea Datelor(LDD) (Data Definition Languages –DDL)

●Limbaje pentru Manipularea Datelor (LMD) (Data Manipulation Languages–DML)

## 1.2 Arhitectura interna a unui Sistem de Baze de Date

Arhitectura pe 3 niveluri relativ independente: nivelul intern, nivelul conceptual şi nivelul extern(Standard ANSI/X3/SPARC -1975)

􀂄Schema 􀃆descrierea datelor pe un anumit nivel: schema interna, conceptuala si scheme externe (vedere utilizator)

􀂄Corespondente intre niveluri (mappings)

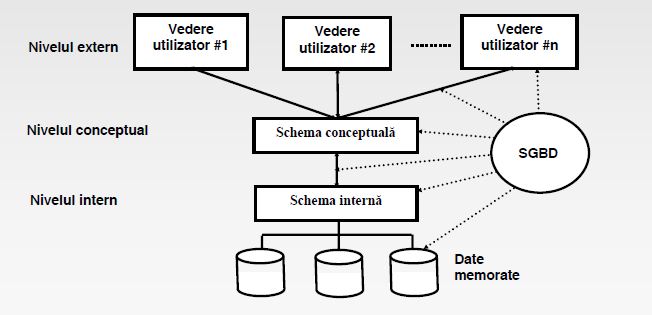


Figura 1. 2 Arhitectura unui sistem de baze de date

## 1.3 Avantaje oferite de Sistemele de BD

Compactitate ridicatăa datelor

􀂄Reprezentarea unor asocieri complexe intre date

􀂄Timp de dezvoltare a bazelor de date redus

􀂄Viteza mare de actualizare si regasire a datelor

􀂄Redundanta controlata a datelor (si cat mai scazuta)

􀂄Flexibilitate, mentinerea datelor actualizate la zi

􀂄Independenta datelor fata de suportul hardware utilizat

􀂄Securitatea datelor: autentificarea utilizatorilor si autorizarea accesului

􀂄Impunerea de restrictii (constrangeri) de integritate la introducerea si actualizarea datelor

􀂄Mentinerea integritatii datelor in caz de defecte: salvare si refacere

􀂄Posibilitatea de partajare a datelor intre mai multe categorii de utilizatori

􀂄Posibilitatea de introducere a standardelor

## 1.4 Clasificarea Sistemelor de Baze de Date

Clasificare dupa modelul de date:

●Modelul ierarhic de date

●Modelul de date retea

●Modelul relational

●Modelul obiect-orientat

●Modelul obiect-relational

Clasificare dupa numarul de utilizatori

●Sisteme mono-utilizator

●Sisteme multi-utilizator

Clasificare dupa numarul de statii pe care este memorata baza dedate:

●Baze de date centralizate

●Baze de date distribuite

Arhitectura client-server:

●Server (back-end): SGBD-ul si baza de date

●Client (front-end): program (programe) de aplicatie

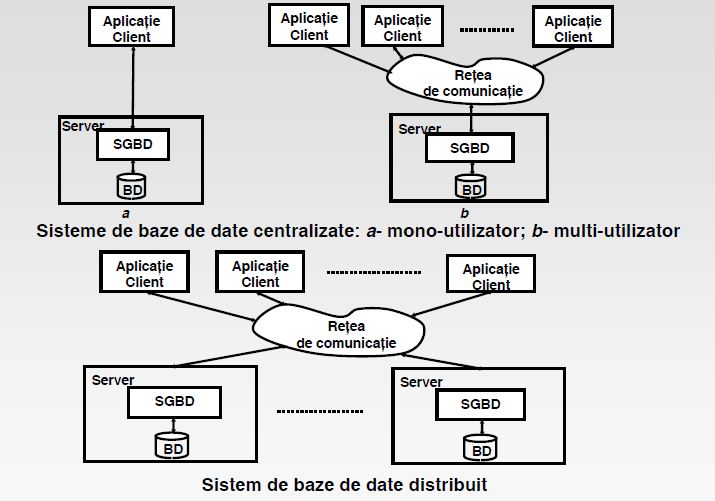


Figura 1. 3 Clasificarea sistemelor de baze de date

## 1.5 Modelarea Datelor

Un model este o abstractizare a unui sistem:

●captează cele mai importante trăsături caracteristice ale sistemului (concepte)

●conceptele trebuie sa fie relevante din punct de vedere al scopului pentru care se defineşte modelul respectiv

􀂄Tehnica de identificare a trăsăturilor caracteristice esenţiale ale unui sistem se numeşte abstractizare

􀂄Un model de datestabileşte regulile de organizare şi interpretare a unei colecţii de date.

􀂄În proiectarea bazelor de date se folosesc2 categorii de modele:

●Modele conceptualede nivel înalt(modelul Entitate-Asociere, modelul Entitate-Asociere Extins)– descriu concis colectiile de date care modelează activitatea dorită fără să detalieze modul de reprezentare sau de prelucrare a datelor -schemă conceptuală de nivel înalt

●Modele specializate(modelul ierarhic, modelul reţea, modelul relaţional, etc.) -descriu reprezentarea mulţimilor de entităţi şi a asocierilor dintre acestea prin structuri de date specifice -schemă conceptuală (logică)

􀂄Trecerea de la modelul conceptual de nivel înalt la un model de date specific 􀃆 proiectare logică a bazei de date.

## 1.6 Modelul entitate – Asociere

Modelul Entitate-Asociere(Entity-Relationship Model) defineste multimile de entităţi şi asocierile dintre ele, dar nu impune nici un mod specific de structurare şi prelucrare (gestiune) a datelor; Introdus în 1976 de P.S. Chen

􀂄O entitate(entity)este „orice exista in realitatea obiectiva si poate fi identificat în mod distinctiv“

●Exemple: o persoana,o planta, o activitate, un concept etc.

􀂄Un atribut(attribute) este o proprietate care descrie un anumit aspect al unei entităţi

●Exemple: persoanele au nume, prenume, adresa etc.

􀂄Tip de entitate (entity type): se refera la entitătile similare, care pot fi descrise prin aceleasi atribute

●Exemple: tipul persoana, tipul planta

􀂄Multime de entitati(entities set): colecţia tuturor entităţilor de acelaşi tip dintr-o bază de date constituie o mulţime de entităţi

●Exemple: multimea tuturor persoanelor, multimea tuturor plantelor

􀂄O entitate este o instanta a unui tip de entitate si un element al multimii de entitati de acel tip

􀂄In exprimarea curenta, adeseori nu se face diferentierea dintre entitate, tip de entitate si multime de entitati, dar diferenta este evidenta

􀂄Asemanare cu modelul obiect: tip de entitate -clasa; entitate –obiect

## 1.7 Asocieri

O asociere(relationship) este o legătură (corespondenţă)între entităţi din două sau mai multe mulţimi de entităţi; asocierile pot avea atribute

􀂄Tipul asocierii(relationship type) –se refera la asocierile similare, care pot fi definite intre entitati din 2 sau mai multe multimi de entitati

􀂄Multime de asocieri(relationship set): multimea asocierilor de acelasi tip

􀂄O asociere este o instanta a unui tip de asociere si un element al multimii de asocieri de acel tip

􀂄In exprimarea curenta, adeseori nu se face diferentierea dintre asociere, tip de asociere si multime de asocieri, dar diferenta este evidenta

􀂄Gradulunui (tip de) asociere (degree): numărul de (mulţimi de) entităţi asociate; dupa grad, asocierile pot fi:

●binare(de gradul 2, între 2 mulţimi de entităţi) –majoritatea asocierilor

●multiple(între k mulţimi de entităţi, k > 2)

􀂄Categorii (tipuri) de asocieri binare - după numărul elementelor din fiecare dintre cele două mulţimi puse în corespondenţă:

●“unul-la-unul”(one-to-one) –1:1; exemplu: sot-sotie

●”unul-la-multe”(one-to-many) –1:N; exemplu: parinte-fii

●“multe-la-unul”(many-to-one) –N:1; exemplu: fii-parinte

●“multe-la-multe”(many-to-many) –M:N; exemplu: profesori-studenti

## 1.8 Categorii de asocieri binare

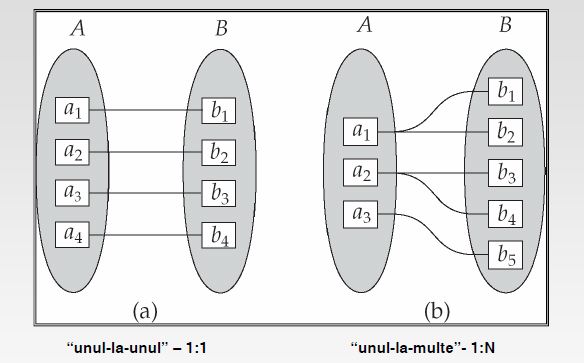


Figura 1. 4 Asocieri unu la unu și unu la multe

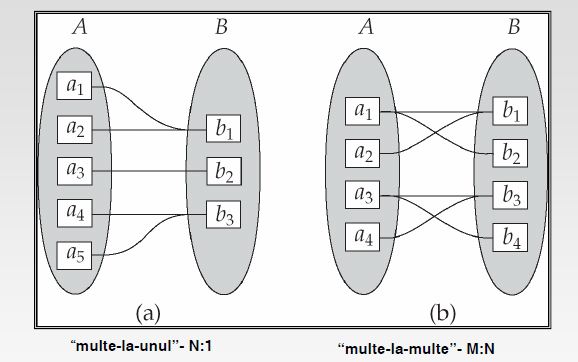


Figura 1. 5 Asocieri multe la unu și multe la multe

## 1.9 Cardinalitatea asocierilor

Cardinalitatea (multiplicitatea) unei asocieri faţă de o mulţime de entităţi(cardinality, multiplicity) este numărul maxim de elemente din acea mulţime care pot fi asociate cu un element din altă mulţime a asocierii

●Exemplu: asocierea “unul-la-multe”dintre mulţimile A şi B prezintă multiplicitatea 1 faţă de mulţimea A şi multiplicitatea N (se înţelege o valoare oarecare N > 1) faţă de mulţimea B

􀂄Raport de cardinalitate(cardinality ratio): raportul dintre valorile cardinalităţilor unei asocieri faţă de două din mulţimile de entităţi asociate

●Exemple pentru asocieri binare: 1:1, 1:N, N:1, M:N

●Asocierile multiple (k-are, k > 2) prezintă câte un raport de cardinalitate pentru fiecare pereche de mulţimi de entităţi pe care le asociază.

## 1.10 Diagrama Entitate-Asociere

􀂄Diagrama Entitate-Asociere(Entity-Relationship Diagram) reprezintă grafic modelul Entitate-Asociere prin mulţimile de entităţi şi asocierile dintre acestea

􀂄Multimi (tipuri) de entitati:

●Puternice (de sine statatoare)

●Slabe (depind de alte multimi de entitati)

􀂄Notatii:

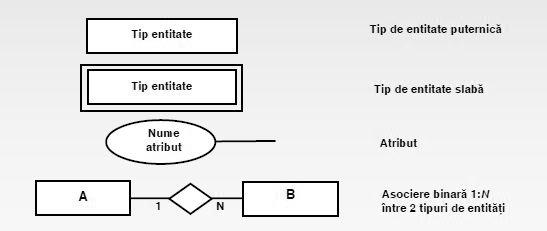


Figura 1. 6 Diagrama entitate-asociere

Exemplu:

Multimi de entitati puternice:

●SECTII (Numar, Nume, Buget)

●ANGAJATI (Nume, Prenume, DataNasterii, Adresa, Functie, Salariu)

●PROIECTE (Denumire, DataInceperii, Termen, Buget)

􀂄Multimi de entitati slabe: DEPENDENTI (Nume, Prenume, DataNasterii,GradRud)

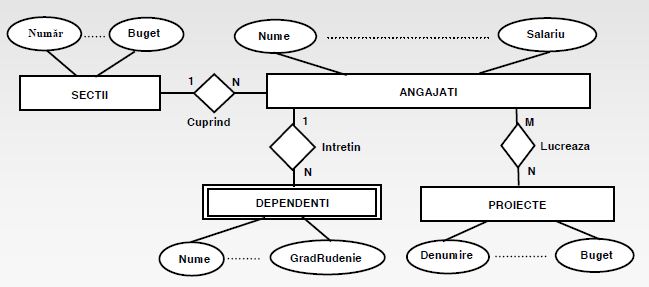


Figura 1. 7 Exemplu de diagramă entitate-asociere

**Asocieri**:

●Asocierea SECTII -ANGAJATI -1:N

●Asocierea ANGAJATI -PROIECTE -M:N

●Asocierea ANGAJATI -DEPENDENTI -1:N

􀂄Raportul de cardinalitate al unei asocieri este stabilit de proiectantastfel încât să reflecte cât mai corect modul de organizare a activităţii modelate

􀂄Modul de stabilire a tipurilor de entităţi şi a asocierilor nu este unic: aceeaşi funcţionalitate se poate obţine printr-o varietate de diagrame E-A

􀂄O mulţime de entităţi se denumeste printr-un substantiv, iar o asociere se denumeste (de regula) printr-un verb, deoarece o asociere reprezintă o interacţiune între entităţi

􀂄Modelul E-A nu precizează modul cum sunt realizate asocierile între mulţimile de entităţi: acest aspect depinde de modelul de date specializat utilizat pentru definirea bazei de date

●Exemple: în modelul ierarhic asocierile sunt realizate explicit, prin pointeri de la o entitate la entităţile asociate; în modelul relaţional asocierile se realizează prin egalitatea valorilor unor atribute comune ale multimilor de entităţi (chei)

## 1.11 Modelul Entitate – Asociere Extins

Modelul Entitate-Asociere Extins (Enhanced Entity-Relationship Model) permite definirea de subtipuri ale unui tip de entităţi, care moştenesc atribute de la tipul de entitate respectiv

􀂄Crearea ierarhiilor: specializaresi generalizare

􀂄Tipurile şi a subtipurile formeaza ierarhii de tipuri de entităţi complexe, organizate pe mai multe niveluri

􀂄Diagrama Entitate-Asociere Extinsa

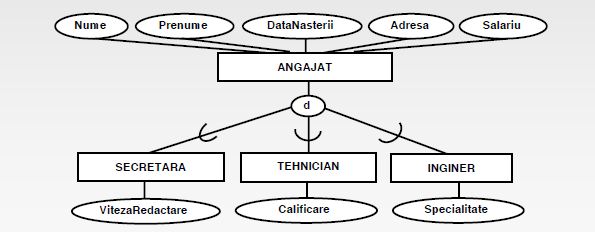


Figura 1. 8 Modelul entitate-asociere extins

## 1.12 Modelul de date ierarhic

􀂄Modelul ierarhic(Hierarchical Model): baza de date se reprezinta printr-o structură ierarhică de înregistrări (records) conectate prin legături (links).

●A fost primul model folosit pentru dezvoltarea bazelor de date

●Cel mai cunoscut SGBD ierarhic: sistemul IMS (Information Management System) dezvoltat de IBM în programul de cercetări Apollo, în perioada anilor 1960

􀂄O înregistrare de dateîn modelul ierarhic este o instanţă a unui tip de înregistrare (record type) şi constă dintr-o colecţie de câmpuri (fields), fiecare câmp conţinând valoarea unui atribut.

􀂄Un tip de legătură în modelul ierarhic: tip de asociere cu raportul de cardinalitate 1:N (părinte-fiu) între două tipuri de înregistrări

􀂄Schema conceptuală a unei baze de date în modelul ierarhic se reprezintă printr-un număr oarecare de scheme ierarhice

􀂄O schemă ierarhică este un arbore direcţionat, reprezentat pe mai multe niveluri, în care nodurile sunt tipuri de înregistrări, iar arcele sunt tipuri de legături

## 1.13 Baze de date ierarhice

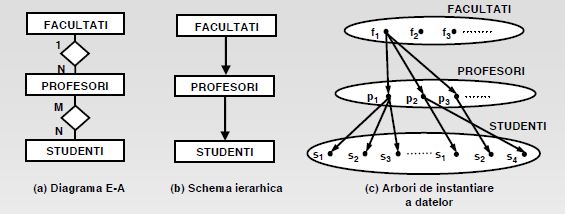


Figura 1. 9 Modelul ierarhic

􀂄Numai legături de tipul părinte-fiu, care corespund asocierilor 1:1 şi 1:N din modelul E-A

􀂄Asocierile M:N se pot reprezenta prin multiplicarea înregistrărilor de tip fiu, atunci când sunt referite de mai multe înregistrări de tip părinte

●mare redundanţă a datelor

􀂄Avantaje: simplitatea şi eficienţa de calcul

􀂄Deficiente:

●nu exista separare intre descrierea logica si fizica a datelor

●interogarile trebuie să fie prevăzute explicit in structura datelor

􀂄Utilizari actuale -aplicatii specializate, baze de date XML

## 1.14 Modelul de date retea

􀂄Modelul reţea(Network Model) foloseşte o structură de graf pentru definirea schemei conceptuale a bazei de date

􀂄Modelele ierarhic si retea

􀃆modele pre-relationale

􀂄Standardizat în 1971, de o comisie DBTG (Database Task Group).

􀂄Sisteme de gestiune comerciale in modelul retea: IDS II (Honeywell), UNISYS (Burroughs), IDMS (Computer Associates)

􀂄Nodurile grafului sunt tipuri de entităţi (înregistrări -records), iar muchiile reprezintă asocierile (legăturile-links) dintre tipurile de entităţi

􀂄Asocierile M:N se reprezintă fără duplicarea înregistrărilor, fiecare înregistrare putând fi referită de mai multe înregistrări, ceea ce elimină (micșorează) redundanţa

􀂄Dezavantaje:

􀂄aceleasi ca si la modelul ierarhic, la care se adauga

􀂄complexitatea mare in reprezentarea datelor􀂄Actualmente modelul retea:

􀂄este rar utilizat pentru baze de date de uz general

􀂄se utilizeaza pentru aplicații specializate de ex, pentru baze de date grafice (scene virtuale)

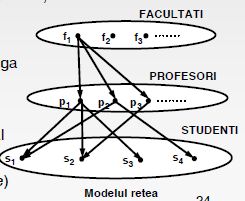


Figura 1. 10 Modelul rețea

## 1.15 Modelul de date relational

􀂄Modelul relaţional(Relational Model) se bazează pe noţiunea de relaţie(relation) din matematică, care corespunde unei mulţimi de entităţi

􀂄Fundamentat de E.F. Codd (IBM), prin lucrarea "Un model Relaţional de Date pentru Bănci Mari de Date Partajate" (1970)

􀂄Dezvoltare extraordinara a sistemelor de gestiune a bazelor de date relationale, datorită simplităţii şi a fundamentării matematice a modelului

􀂄Alte lucrări ale cercetatorilor C.J. Date, P. Chen, R. Boyce, J.D. Ullman, R. Fagin, W. Armstrong, M. Stonebraker etc. au perfecţionat modelul relaţional

􀂄Primul Sistem de Gestiune a Bazelor de Date Relaţionale (SGBDR) a fost prototipul System R (IBM, 1970)

􀂄După aceasta numeroase companii au realizat sisteme de gestiune relaţionale: Oracle, Microsoft, Ingres, Sybase, IBM, Informix

􀂄SGBDR folosesc limbajul SQL (Structured Query Language), pentru care au fost emise mai multe standarde ANSI (American National Standardization Institute) si ISO (International StandardizationOffice)

􀂄Majoritatea SGBD-urilor relaţionale actuale implementează versiunea SQL2 (SQL92) sau versiuni ulterioare (SQL-1999, SQL-2003, SQL-2006)

## 1.16 Modelul obiect obiect-orientat

􀂄Modelul obiect(Object Model) este un concept unificator

􀂄Necesar in domenii in care se manipuleaza date de tipuri complexe:

●proiectarea sistemelor de calcul: programare, hardware, interfete

●proiectarea asistată de calculator (CAD-CAM)

●sisteme de informaţii geografice

●fizică, biologie, medicină şi altele

􀂄Strategii pentru dezvoltarea sistemelor de gestiune a bazelor dedate obiect-orientate (SGBDOO):

●Extinderea unui limbaj de programare obiect-orientat cu capacităţi de administrare a obiectelor persistente: sistemul GemStone (extinde Java si C++)

●Extinderea unui limbaj de programare relaţional cu capacităţi de orientare spre obiecte. Exemplu: limbajul OQL (Object Query Language) (sau Object SQL), Există mai multe astfel de sisteme, cum sunt: Ontos, Versant, O2.

●Dezvoltarea unui limbaj obiect-orientat pentru baze de date complet nou: SIM (Semantic Information Manager).

􀂄Dificultati:

●Complexitate in dezvoltare a bazei de date şi a aplicaţiilor

●Interogarile trebuie să fie prevăzute explicit in structura datelor

􀂄Utilizare SGBDOO: cam 5% din sistemele de gestiune actuale

Modelul obiect obiect-relational:

􀂄Modelul obiect-relaţional(Object-Relational Model) reprezintă extinderea modelului relaţional cu caracteristici ale modelului obiect

􀂄Modelul obiect-relaţional păstrează structurarea datelor în relaţii, si, in plus:

●permite definirea unor noi tipuri de date, ca domenii ale atributelor

●permite extinderea tipurilor de date prin moştenire

􀂄Sistemele de gestiune a bazelor de date obiect-relaţionale (SGBDOR) se realizează prin extinderea sistemelor relaţionale, de regula în mod gradat, adăugându-se de la o versiune la alta cât mai multe caracteristici posibile ale modelului obiect

􀂄Aceasta abordare asigură rularea în continuare a aplicaţiilor relaţionale existente în noile versiuni de sisteme SGBDOR, ceea ce permite producătorilor să-şi păstreze clienţii şi domeniile de utilizare

􀂄Limbajele de programare pentru SGBDOR sunt implementări de standarde mai recente ale limbajului SQL: SQL3 (SQL-1999), SQL-2003, SQL-2006

## 1.17 Complexitatea datelor si a interogarilor

Clasificare propusa de M. Stonebraker (1996)



Figura 1. 11 Grafic cu complexitatea datelor și a interogărilor

􀂄SGBDR prelucrează tipuri simple de date, dar permit interogări complexe

􀂄SGBDOO prelucrează tipuri de date complexe, dar în care rezolvarea interogărilor este destul de dificilă

􀂄SGBDOR permit prelucrarea datelor complexe şi rezolvarea interogărilor complexe; sistemele de baze de date obiect-relaţionale sunt considerate sisteme de baze de date universale

## 1.18 Evolutia sistemelor de baze de date

**1960** Modele prerelationale: ierarhic si retea

Primele produse de baze de date (DBOM, IMS, IDS, Total, IDMS)

Standarde Codasyl

**1970**  Modelul relational –prototipuri de SGBDR

Lucrari teoretice asupra modelului relational

Arhitectura interna pe 3 niveluri a bazelor de date (ANSI and Codasyl)

Modelul Entitate-Asociere

**1980**  Dezvoltarea SGBDR comerciale

Primul standard SQL (1986 -ANSI, ISO)

Baze de date distribuite

**1990** Arhitectura client/server a sistemelor de baze de date (two-tier arch.)

Baze de date obiect-orientate

Baze de date obiect-relationale

Standarde SQL: SQL 92, SQL 99

**2000** Arhitectura pe 3 niveluri a sistemelor de baze de date (three-tier arch.)

Baze de date in sistemul WWW (World Wide Web)

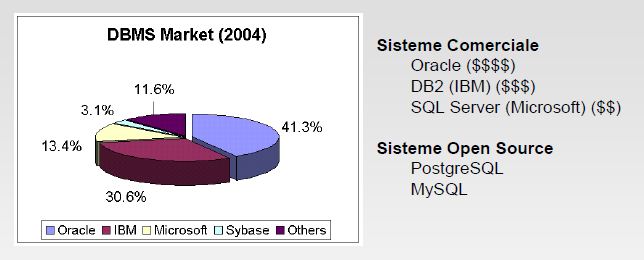


Figura 1. 12 Sisteme de gestiune a bazelor de date

# Capitolul II – BAZE DE DATE RELAȚIONALE

## 2.1 Bază de Date Relaționale. Definiție. Relații. Atribute. Domenii

### 2.1.1 BD – Definiție

O bază de date relaţională este o bază de date care respectă modelul relaţional. Modelul de date Relaţional (Relational Model) se bazează pe noţiunea de relaţie din matematică, care corespunde unei entităţi de acelaşi tip şi are o reprezentare uşor de înţeles şi de manipulat, ce constă dintr-un tabel bidimensional, compus din linii şi coloane. Fiecare linie din tabel reprezintă o entitate şi este compusă din mulţimea valorilor atributelor entităţii respective, fiecare atribut corespunzând unei coloane a tabelului.

O bază de date relaţională este compusă dintr-o mulţime finită de relaţii, având următoarele caracteristici:

* fiecare relaţie reprezintă o mulţime (tip) de entităţi sau o mulţime (tip) de asocieri;
* fiecare relaţie este unică într-o bază de date;
* relaţia se defineşte prin intermediul atributelor sale.

### 2.1.2 Atribute. Relații. Domenii

Atributele unei relaţii corespund atributelor tipului de entitate sau de asociere pe care îl reprezintă relaţia respectivă, astfel:

* fiecare atribut are un nume (Ai) şi un domeniu de definiţie D(Ai);
* pentru o entitate dată, un atribut poate lua o singură valoare.

Atributele pot fi: simple (un element) sau compuse (o submultime de atribute). Domeniul reprezintă o mulțime de valori D = {di| i = 1,…, n }, definit printr-o specificare de tip, unde:

* D este numele domeniului;
* di este un element al domeniului care satisface anumite constrangeri.

Elementele domeniilor sunt atomice (indivizibile) și există o valoare speciala, null, care aparține oricărui domeniu (valoare necunoscută).

O relaţiei este descrisă prin schema relației: R(A1,A2,...Ai,...An), unde:

* R este numele schemei relaţiei;
* A1,A2,...Ai,..An este lista ordonată a atributelor sale;
* fiecare atribut Ai este definit pe domeniul său de definiţie, D(Ai).

Gradul relaţiei este dat de numărul de atribute ale schemei acelei relatii (n); de exemplu STUDENȚI (Nume, Prenume, DataNașterii, Adresa, Facultatea). O relaţie r definită prin schema R(A1,A2,...Ai,...An) este:

* mulţime finită de n-tupluri t;
* tuplul t este o listă ordonată de n valori: t = <v1,v2,...vi,...vn>, unde 1 ≤i ≤n;
* vi este o valoare a atributului Ai, vi ∈D(Ai).

Pentru a reprezenta grafic o relaţie se utilizează tabelul (table); compus din (în figură se poate vedea tabelul corespunzător relației STUDENȚI:

* Numele tabelului – identic cu numele relaţiei;
* Coloanele – corespunde atributelor relației;
* Capul tabelului – conține numele atributelor (coloanelor);
* Omulţime de linii, fiecare linie corespunzând unui tuplu;
* Valori ale atributelor fiecarui tuplu.

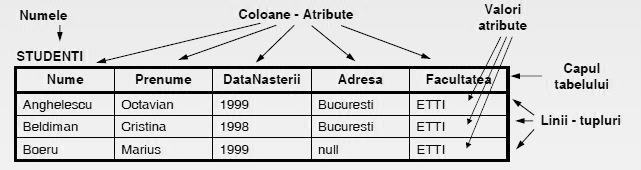


Figura 2. 1 Model de tabel creat

Din punct de vedere logic, ordinea valorilor atributelor într-un tuplu nu contează; această structurare poate fi exprimată prin următoarele definiţii:

* Schema relaţiei: R = {A1,A2, ...Ai,...An} (o mulţime de atribute);
* Relaţia r(R): o mulţime de n-tupluri t, unde:
* fiecare tuplu t este o mulţime de n perechi ordonate <Ai,vi>,unde 1 ≤i ≤n;
* t = {<A1,v1>,<A2,v2>,...<Ai,vi>, ...<An,vn>};
* vi este valoarea atributului Ai, vi ∈D(Ai).

## 2.2 Limbajul SQL

### 2.2.1 Caracteristici Generale

Majoritatea sistemelor relaționale suportă diferite variante (dialecte) ale limbajului SQL (Structured Query Language). Limbajul SQL a fost dezvoltat într-un prototip de sistem relațional -System R - la compania IBM la mijlocul anilor 1970. În anul 1979 corporatia Oracle a introdus prima implementare a limbajului SQL în varianta comercială. În anul 1986 Institutul Național American de Standarde (ANSI) a definit standardul limbajului SQL pentru bazele de date relaționale. Organizația Internațională de Standarde (ISO) a adoptat de asemenea SQL ca limbaj standard pentru sistemele relțtionale, sub denumirea de SQL-92 (sau, mai simplu, SQL2).

Limbajul SQL înglobeaza mai multe componente, dintre care cele mai importante sunt:

* Componenta de descriere a datelor (LDD - Limbaj de Descriere a Datelor) (Data Description Language - DDL);
* Componenta de manipulare a datelor (LMD - Limbaj de Manipulare a Datelor) (Data Manipulation Language - DML).

Limbajul SQL folosește reprezentarea prin tabele a relaţiilor, reprezentare care este mai simplă şi mai intuitivă (foloseste termenii tabel, linie, coloană). Limbajul SQL2 este un limbaj neprocedural:

* instrucţiune SQL2 specifică ce informaţii trebuie să fie setate sau obţinute, nu modul (procedura) în care se operează:
* limbajul SQL2 nu conţine instrucţiuni de control al fluxului execuţiei (instrucţiuni ca for, while, if, etc).

Standardul SQL3 prevede instrucțiuni de control și crearea de tipuri definite de utilizator, fiind implementat în SGBD-urile obiect-relaționale. Pentru aplicaţiile de baze de date, s-au dezvoltat extensii procedurale ale limbajului SQL, biblioteci şi interfeţe de programare care integrează instrucţiunile SQL.

### 2.2.2 Structura Lexicală. Expresii și Operatori

O instrucţiune SQL (statement) este o secvenţă de elemente - de regula terminată cu semnul punct şi virgulă (;). Fiecare instrucţiune SQL conţine o comandă SQL (command), care specifică ce acţiune se efectuează, urmată de alte elemente, care specifică operaţii, clauze, parametri, etc; de exemplu: SELECT \* FROM ANGAJATI.

Elementele (tokens) instrucţiunilor SQL sunt:

* Cuvinte cheie (key words): CREATE, INSERT, SELECT, WHERE, FROM, etc;
* Identificatori (identifiers):
* simpli – caractere alfa-numerice si underscore(\_): ANGAJATI, Nume, Prenume;
* delimitati (quoted) – pot conține orice caracter, folosește ghilimele: ‘Nume’, ‘Prenume’;
* constante (literali): 1000, 100.5, ‘Ionescu’, NULL;
* caractere speciale: \* , . .

Spaţiile albe (whitespaces) separa elementele: spaţiu, linie nouă, tab. O instructiune se poate scrie pe una sau mai multe linii, iar într-o linie se pot introduce una sau mai multe instructiuni. Limbajul SQL este case-insensitive (nu deosebește literele mici de cele mari) cu excepția identificatorilor delimitați (quoted) care sunt case-sensitive.

O expresie SQL constă dintr-unul sau mai mulţi operanzi, operatori şi paranteze. Parantezele se pot folosi pentru a preciza o anumită ordine a operaţiilor, dacă aceasta este diferită de ordinea implicită data de precedenta operatorilor. Un operand poate fi:

* numele unei coloane – se folosește valoarea memorată în acea colona într-una sau mai multe linii ale tabelului;
* constantă (literal);
* valoarea returnată de o funcție.

Un operator SQL este exprimat prin:

* unul sau mai mai multe caractere speciale: +, -, \*, /, %, <= etc;
* un cuvânt cheie; exemple: AND, OR, NOT, LIKE.

Operatori SQL se mai pot clasifica după numărul de operanzi: binari sau unari, sau în funcție de rolul avut: aritmetici, logici, de comparaţie SQL, relaționali. Dintre aceste subcategorii menționăm:

* Operatori aritmetici de operații cu numere intregi sau reale: +,-, \*, /, %, ^;
* Operatori aritmetici orientați pe biți: ~, &, |, #;
* Operatori aritmetici de comparație: <, >, =, <> (sau !=), <=, >=;
* Operatori de comparație SQL: IS NULL, IS NOT NULL, BETWEEN, IN, LIKE;
* Operatori relaționali: UNION, INTERSECT, MINUS.

Operatorii de comparaţie returnează valori logice:

* true (1), dacă condiţia este îndeplinită;
* false (0) dacă condiţia nu este îndeplinită;
* null dacă ambii operanzi au valoarea null.

Operatorii logici (NOT, AND, OR) se aplică unor valori logice trivalente (cu 3 valori: true (1), false (0) şi null - lipsa de informație) și returnează o valoare logică trivalentă.

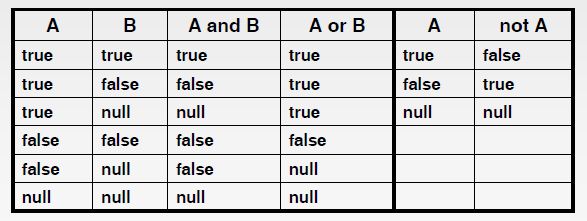


Figura 2. 2 Operatori logici

### 2.2.3 Funcții și Tipuri de Date SQL

Funcţiile SQL sunt de 2 tipuri: funcţii agregat şi funcţii scalare. Funcţiile agregat calculează un rezultat din mai multe linii ale unuitabel (aceste funcţii vor fi detaliate ulterior, la descrierea instrucţiunii SELECT).

Funcţiile scalare primesc unul sau mai multe argumente şi returnează valoarea calculată sau NULL în caz de eroare. Argumentele acestor funcţiilor pot fi constante (literale) sau valori ale atributelor specificate prin numele coloanelor corespunzatoare. Funcţiile scalare se folosesc în expresii, care pot să apară în diferite clauze ale instrucţiunilor SQL. Există 3 clase de funcții scalare:

* Funcţii de calcul trigonometric (sin, cos, tanetc.), funcţii de calcul al logaritmului (ln, log), al puterii (power), funcţii de rotunjire (floor, ceil);
* Funcţii pentru manipularea şirurilor de caractere: concat, replace, upperetc;
* Funcţii pentru data calendaristică şi timp: add\_months, next\_day, last\_day etc;
* Funcţii de conversie: to\_number, to\_charetc.

SQL se caracterizează printr-o varietate de tipuri de date: numeric, şiruri de caractere, şiruri de biţi, data (calendaristică), timp. În continuare se va clasifica fiecare tip de date.

1. Tipul numeric

* numere întregi:integersauint(4 octeţi), smallint(2 octeţi);
* numere reale reprezentate în virgulă flotanta: float (4 octeţi), realşi double [precision](8 octeţi);
* numere zecimale reprezentate cu precizia dorită (tipul numeric sau decimal, memorate ca şir de caractere): numeric[(p,s)](sau decimal [(p,s)]), unde p (precizia) este numărul total de cifre, iar s (scara) este numărul de cifre după punctul zecimal.

2. Șiruri de caractere

* character(n),prescurtat, char(n) - şir de caractere de lungime fixă (n);
* character varying(n),prescurtat varchar(n) - şir de caractere de lungime variabilă, maximum n

3.Șiruri de biţi - secvenţe de cifre binare (care pot lua valoarea 0 sau 1):

* bit(n) - șir de biti de lungime fixă (n);
* bit varying(n) șir de biti lungime variabilă, maxim n;

4. Dată calendaristică şi timp sunt: date, time, timestamp, interval:

* Tipul date: memorarea datelor calendaristice prin utilizarea a trei câmpuri (year, month, day), în formatul yyyy-mm-dd; se admit numai date valide;
* Tipul time: memorarea timpului, folosind trei câmpuri (hour, minute, second) în formatul HH:MM:SS; se admit numai valori valide;
* Tipul timestamp(p): memorarea combinată a datei calendaristice şi a timpului, cu precizia p pentru câmpul second. Valoarea implicită a lui p este 6;
* Tipul intervaleste utilizat pentru memorarea intervalelor de timp.

Standardul SQL2 nu suportă tipuri de date şi operaţii definite de utilizator. Standardul SQL3 suportă tipuri de date şi operaţii definite de utilizator, care sunt caracteristice ale modelului de date obiect-relaţional.

### 2.2.4 Domenii și Instrucțiuni SQL

In SQL2 domeniile atributelor se specifică pe baza tipurilor de date predefinite ale limbajului SQL, deci nu corespund intru totul cunoţiunea de domeniu relaţional. Standardul SQL2 prevede comanda CREATE DOMAIN, care defineste un domeniu pe baza unui tip predefinit SQL2 şi cu unele constrângeri. Standardul SQL3 prevede comanda CREATE TYPE care creaza tipuri definite de utilizator (user-defined types).

În Oracle (8i, 9i, 10g,11g, 18.c) se pot crea tipuri de date noi, folosind comanda CREATE TYPE, care permite gruparea sub un anumit nume a mai multor atribute si operatii.

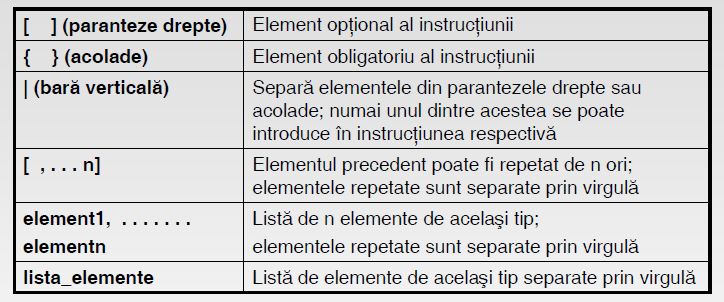


Figura 2. 3 Structura comenzilor in Oracle

Componenta de definire a datelor din SQL (LDD -Limbajul de Definire a Datelor) implică următoarele comenzi:

* Crearea (CREATE), modificarea (ALTER) şi distrugerea (DROP) obiectelor bazei de date;
* Obiectele bazei de date sunt: tabele de bază (TABLE), tabele vedere (VIEW), indexuri (INDEX), proceduri (PROCEDURE), trigere (TRIGGER), utilizatori (USER).

În continuare se vor prezenta exemple de comenzi SQL de definire a datelor:

CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE USER

CREATE FUNCTION, CREATE TRIGGER, CREATE PROCEDURE

ALTER TABLE, ALTER VIEW, ALTER FUNCTION, ALTER PROCEDURE

DROP TABLE, DROP VIEW, DROP INDEX, DROP USER

DROP FUNCTION, DROP PROCEDURE, DROP TRIGGER

Componenta de manipulare a datelor din limbajul SQL (Limbajul deManipulare a Datelor -LMD) conţine comenzile: SELECT, INSERT, UPDATE si DELETE

Instrucțiunile SQL se transmit SGBD-ului de către diferite programe client (client grafic, linie de comandă, program executabil), SGBD-ul compilează și execută instrucțiunea SQL și returnează un raspuns (rezultatul operației sau un cod de eroare).

# CAPITOLUL III: PROIECTAREA ȘI IMPEMENTAREA BAZEI DE DATE

## 3.1 Relația entitate – atribut

Datele cuprinse de catre abonați și de către clienții prepay sunt reprezentate de tabelele: Adresa, Client, Factura, Telefon, Contract, Subscriptie, Prepay, Abonament, Cost și Servicii. Pentru a înțelege mai bine tabelele acestea vor fi detaliate după cum urmează:

* Client – reprezintă tabela de identificare atât a abonaților cât și a clienșilor pre-pay ce conține date precum: nume, prenume și idAdresa;
* Adresa – reprezintă tabela în care sunt înregistrate datele referitoare la reședința clientului, date precum: orașul, strada și numărul;
* Factura – reprezintă tabela ce reține infromații referitoare la suma pe care un client o are de plată precum și data emiterii și scadentă a facturii;
* Contractul – reprezintă tabela ce reține informațiile legate de natura contractului dintre operator și client, date precum: data începrere contract, data sfârșire contract și suma pe care o are de platit lunar. În cazul clienților ce opteaza pentur cartelepe pre-pay atât datele contractuale cât și cele ale facturii vor coincide;
* Telefon – reprezintă tabela în care sunt înregistrate numerele de telefon alocate de către operator fiecărui client în parte;
* Abonament – reprezintă tabela ce conține înregistrari precum tipul abonamentului și starea financiara (”la zi” sau ”întârziat”);
* Prepay – reprezintă tabelul ce conține informații precum creditul de care mai dispune clientul;
* Costul – reprezintă tabela ce reține informații despre costurile suplimentare pe care le efectuiază clientul la pachetele precum: voce, sms, mms, call, ș.a.m.d;
* Serviciul – reprezintă tabela ce reține informații despre tipurile de pachete pe care le are activate în acest moment clientul, pachete precum: voce, sms, mms, call, date, ș.a.m.d;



Figura 3. 1 Diagrama Entitate-Asociere tip1

Diagrama Entitate – Asociere se poate construi pornind de la Figura 5-4 și adăugând următoarele relații între tabele:

Relații de tip ”one to one” (unul la unul):

* Tabela Factura cu tabela Contract
* Tabela Contract cu tabela Subscriptie
* Tabela Contract cu tabela Telefon
* Tabela Telefon cu tabela Subscriptie
* Tabela Cost cu tabela Subscriptie
* Tabela Prepay cu tabela Subscriptie
* Tabela Abonament cu tabela Subscriptie
* Tabela Prepay cu tabela Cost
* Tabela Prepay cu tabela Serviciu
* Tabela Abonament cu tabela Cost
* Tabela Abonament cu Serviciu

Relații de tip ”one to many” (unul la mulți):

* Tabela Adresa cu tabela Client
* Tabela Client cu tabela Subscriptie

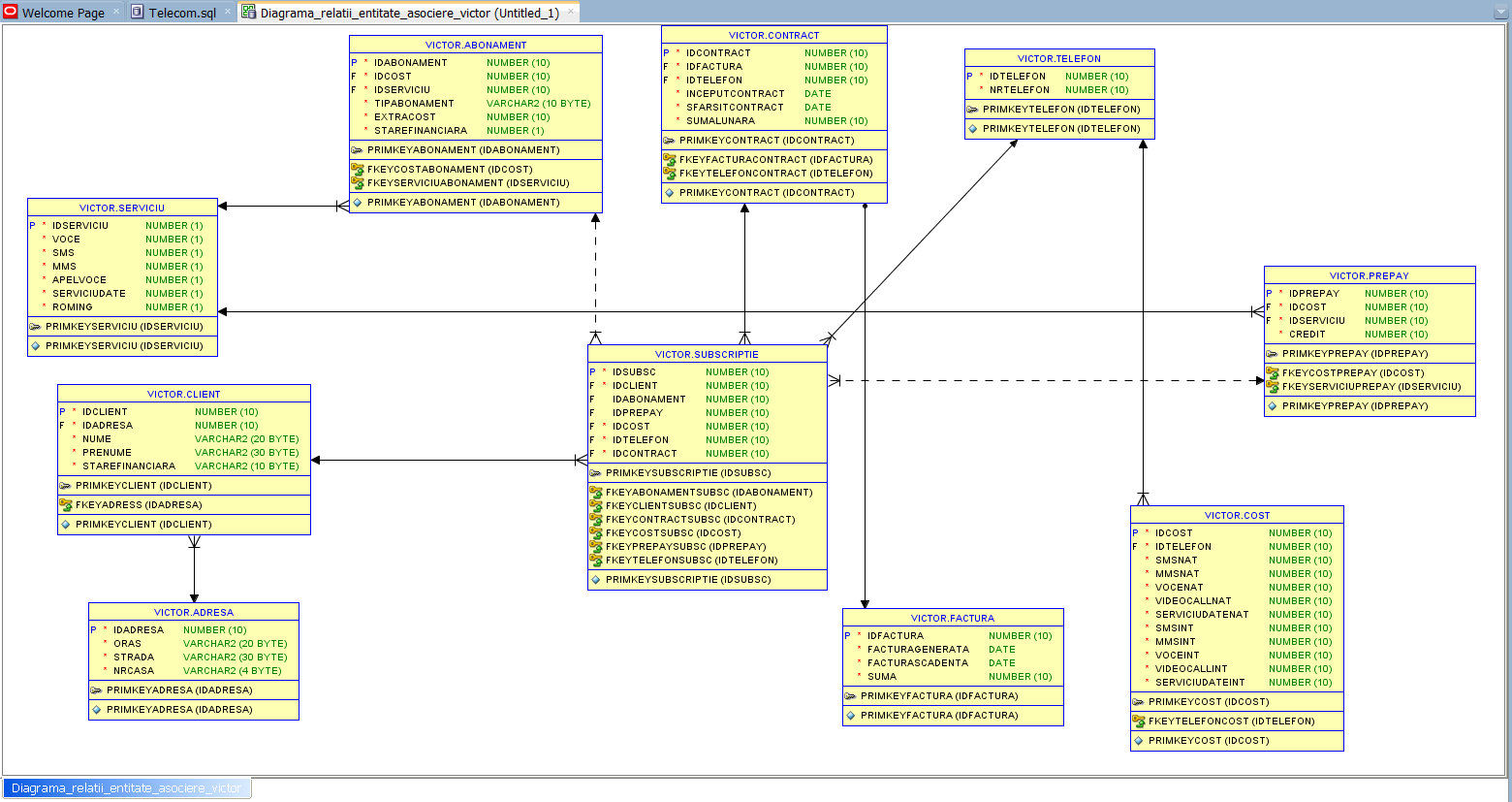


Figura 3. 2 Diagrama Entitate-Asociere tip2

|  |  |  |
| --- | --- | --- |
| **Entitate** | **Atribut** | **Tip Atribut** |
| COST | IDCOST | NUMBER(10,0) |
| IDTELEFON | NUMBER(10,0) |
| SMSNAT | NUMBER(10,0) |
| MMSNAT | NUMBER(10,0) |
| VOCENAT | NUMBER(10,0) |
| VIDEOCALLNAT | NUMBER(10,0) |
| SERVICIUDATENAT | NUMBER(10,0) |
| SMSINT | NUMBER(10,0) |
| MMSINT | NUMBER(10,0) |
| VOCEINT | NUMBER(10,0) |
| VIDEOCALLINT | NUMBER(10,0) |
| SERVICIUDATEINT | NUMBER(10,0) |
| ABONAMENT | IDABONAMENT | NUMBER(10,0) |
| IDCOST | NUMBER(10,0) |
| IDSERVICIU | NUMBER(10,0) |
| TIPABONAMENT | VARCHAR2(10) |
| EXTRACOST | NUMBER(10,0) |
| STAREFINANCIARA | NUMBER(10,0) |
| ADRESA | IDADRESA | NUMBER(10,0) |
| ORAS | VARCHAR2(20) |
| STRADA | VARCHAR2(30) |
| NRCASA | VARCHAR2(4) |
| CLIENT | IDCLIENT | NUMBER(10,0) |
| IDADRESA | NUMBER(10,0) |
| NUME | VARCHAR2(20) |
| PRENUME | VARCHAR2(30) |
| STAREFINANCIARA | VARCHAR2(10) |
| CONTRACT | IDCONTRACT | NUMBER(10,0) |
| IDFACTURA | NUMBER(10,0) |
| IDTELEFON | NUMBER(10,0) |
| INCEPUTCONTRACT | DATE |
| SFARSITCONTRACT | DATE |
| SUMALUNARA | NUMBER(10,0) |
| FACTURA | IDFACTURA | NUMBER(10,0) |
| FACTURAGENERATA | DATE |
| FACTURASCADENTA | DATE |
| SUMA | NUMBER(10,0) |
| PREPAY | IDPREPAY | NUMBER(10,0) |
| IDCOST | NUMBER(10,0) |
| CREDIT | NUMBER(10,0) |
| IDSERVICIU | NUMBER(10,0) |
| SERVICIU | IDSERVICIU | NUMBER(10,0) |
| VOCE | NUMBER(10,0) |
| SMS | NUMBER(10,0) |
| MMS | NUMBER(10,0) |
| VOCECALL | NUMBER(10,0) |
| SERVICIUDATE | NUMBER(10,0) |
| ROMING | NUMBER(10,0) |
| SUBSCRIPTIE | IDSUBSC | NUMBER(10,0) |
| IDCLIENT | NUMBER(10,0) |
| IDABONAMENT | NUMBER(10,0) |
| IDPREPAY | NUMBER(10,0) |
| IDCOST | NUMBER(10,0) |
| IDTELEFON | NUMBER(10,0) |
| IDCONTRACT | NUMBER(10,0) |
| TELEFON | IDTELEFON | NUMBER(10,0) |
| NRTELEFON | NUMBER(10,0) |

Tabel 1 Asocierea entitate – atribute

## 3.2 Modelul entitate - atribut

O entitate este un lucru, obiect, persoană sau eveniment care are semnificaţie pentru afacerea modelată, despre care trebuie să colectăm şi să memorăm date. O entitate poate fi un lucru real, tangibil precum o clădire, o persoană, poate fi o activitate precum o programare sau o operaţie, sau poate fi o noţiune abstractă.

Având în vedere baza de date pe care dorim să o implementăm, toate tabelele menționate anterior reprezintă entități: *Adresa, Factura, Client, Contract, Telefon, Subscriptie, Cost, Serviciu, Prepay, Abonament.*

Un atribut posedă un nume şi - pentru fiecare instanţă a entităţii - poate lua o valoare dintr-o mulţime fixată de valori, numită domeniul de valori ale atributului. Așadar, o entitate este definită de propriile atribute în mod unic, iar un atribut poate fi văzut ca o funcție entitate 🡪 domeniu.

Clasificarea atributelor:

A) Atributele se pot clasifica după complexitate:

* atribute compuse;
* atribute simple sau elementare, după cum ele se mai descompune sau nu în alte atribute, de mai mică complexitate.

Există atribute nu pot fi decât simple (atributele zi, lună, an). Există însă atribute care pot fi considerate fie simple, fie compuse. De exemplu atributul DataScadentă cu valorile: 1 ianuarie 2019, 2 Mai 2019 etc. poate fi privit fie ca un atribut simplu, fie ca unul compus din atributele Zi, Lună, An.

Este indicat să îl tratăm ca un atribut compus, dacă prevedem necesitatea de a avea acces direct la luna sau anul de naştere al unei persoane înregistrate în baza de date. Dacă însă o astfel de necesitate nu este probabilă şi dacă dorim să simplificăm structura entităţii (şi deci a bazei de date), atunci este preferabil să îl tratăm ca atribut simplu.

B) Atribute monovaloare și atribute multivaloare:

* Atributele monovaloare: sunt atributele care, pentru o anumită instanță a unei entități pot lua o singură valoare. Un exemplu în acest caz poate fi data de început a unui contract;
* Atributele multivaloare: sunt acele atribute care pot deține un ansamblu de valori.

C) Atribute stocate și atribute derivate:

* Atributele stocate: sunt acele atribute a căror valoare este salvată efectiv în baza de date;
* Atributele derivate: sunt acele atribute ale căror valori pot fi obținute din atributele stocate și/sau alte informații de sistem. Un exemplu de atribut derivat este dat de atributele *vârstă* și *dataNaștere* asociate unei entități *persoană* deoarece vârsta poate fi dedusă din data nașterii.

D) Atribute cu valoare necunoscută:

* Valoarea nu se dorește sau nu se poate specifica;
* Valoarea nu se cunoaște;
* Nu se aplică. Un exemplu poate fi numărul apartamentului pentru o casă.

Soluția în acest caz este utilizarea valorii NULL.

Pentru un operator de telefonie mobilă, entitățile bazei de date sunt:

* Adresa cu atributele: oras – tip int, oras - tip varchar2 cu dimensiunea maximă de 10 caractere, strada - tip varchar2 cu dimensiunea maximă de 10 caractere, nrCasa- tip varchar2 cu dimensiunea maximă de 10 caractere;
* Factura cu tabela: facturaGenerata – de tip data, facturaScadenta – de tip data, suma – de tip int;
* Client cu tabela: idAdresa – tip int, idSubsc – tip int, nume - tip varchar2 cu dimensiunea maximă de 20 caractere, prenume - tip varchar2 cu dimensiunea maximă de 30 caractere, stareFinanciara - tip varchar2 cu dimensiunea maximă de 10 caractere;
* Contract cu tabela: idFactura – tip int, idTelefon – tip int, startContract – tip data, terminaContract – tip data, sumaLunara – tip int;
* Telefon cu tabela: idSubsc – tip int, nrTelefon – tip int;
* Subscriptie cu tabela: idClient – tip int, idAbonament – tip int, idPrepay – tip int, idCost – tip int, idContract – tip int;
* Cost cu atributele: idTelefon - tip int, smsNat – tip int, mmsNat – tip int, voceNat – tip int, videoCallNat – tip int, serviciuDataNet – tip int, smsInt – tip int, mmsInt – tip int, voceInt – tip int, videoCallInt – tip int, serviciuDateInt – tip int;
* Prepay cu atributele: idCost – tip int, idSubsc – tip int, Credit – tip int, idServiciu – tip int;
* Abonament cu atributele: idCost – tip int, idSubsc – tip int, idServiciu– tip int, tipAbonament - tip varchar2 cu dimensiunea maximă de 10 caractere, extraCost– tip int, stareFinanciara– tip int;
* Serviciu cu atributele: voce– tip int, sms– tip int, mms– tip int, voceCall– tip int, serviciuDate– tip int, roming– tip int.

Tabelul reprezintă o colecție de informații logice relaționale tratate ca o unitate. Acesta constituie practic reprezentarea grafică a unei relații și este compus din înregistrări sau rânduri. Fiecare înregistrare este tratată ca o unitate și se poate elga de înregistrări conținute de alte tabele. Înregistrările sunt constituite din câmpuri sau coloane, ce reprezintă o particulă atomică a bazei de date. Aceasta înseamnă că este cantitatea de informație cea mai mică ce poate fi manipulată. Toate înregistrările dintr-o tabelă au aceleași câmpuri, însă nu toate câmpurile trebuie să fie completate cu date.

## 3.3 Constrangeri

Constrângerile de integritate sunt reguli care se definesc la proiectarea unei baze de date şi care trebuie să fie respectate de-a lungul existenţei acesteia. Entităţiile unei baze de date reflectă realitatea modelată şi de aceea valorile pe care le conţin trebuie să respecte anumite reguli, care să corespundă celor din realitate.

* Constrângeri intra-relație sau în cadrul tabelei – fac referire doar la o singură tabelă din baza de date. Constrângeri de tuplu – pot fi evaluate pentru fiecare tuplu în mod independent de restul celorlalte linii din tabelă. Acestea sunt definite ca expresii condiționale (logice) și sunt formate din atomi care compară valorile unor atribute cu alte atribute sau constante. De asemenea, utilizează operatori de comparație de tipul: *și (AND), sau (OR), negație (NOT).*
* Constrângeri de domeniu – sunt evaluate valorile atributelor pentru a asigura integiratea domeniilor.
  + - Constrângerea NOT NULL – Valoarea NULL este o valoare particulară, care nu reprezintă valoarea 0, ci lipsă de informaţie. Această valoare NULL poate apărea când nu se cunosc respectivele informaţii, ca de exemplu, în cazul bazei de date Subscriptie pentru atributele idPrepay și idAbonament.
    - Constrângerea DEFAULT – Această constrângere este folosită pentru stabilirea unei valori implicite (DEFAULT) pentru un atribut al entităţii. În cazul în care la inserarea unei înregistrări nu se specifică valoarea unui atribut (câmp), atunci acesta primeşte valoarea implicită cum se întâmpla pentru tabela STAREFINANCIARA care are default valoarea ‘la zi’.

## 3.4 Cheile din tabelă

Cheie candidat – un atribut care are valori unice pentru fiecare înregistrare - sau un set de atribute ale căror valori combinate sunt unice, minimală în raport cu această proprietate (nici o submulţime nu e cheie); dacă nu este îndeplinită proprietatea de mulţime minimală avem de a face cu o supercheie

O cheie primară (primary key) este o cheie candidat căreia proiectantul îi oferă un rol special de accesare şi identificare a tuplurilor relaţiei. În plus, se impune ca atributelor cheii primare să nu admită valori de NULL şi să nu fie modificate prin operaţii de actualizare a datelor.

În cazul de față, tributele *idAbonament, idAdresa, idClient, idContract, idCost, idFactura, idPrepay, idServiciu, idSubscriptie, idtelefon* sunt toate chei primare artificiale, special create pentru a manevra cu mai multă uşurinţă tabelele constituente ale bazei de date. Ca si cheie primara naturala ar fi putut fi aleasa cheia nrTelefon din tabela Telefon deoarece fiecare numar de telefon este unic.

|  |  |
| --- | --- |
| **Tabela** | **Cheia primară** |
| Abonament | idAbonament este cheia priamara; TipAbonament, ExtraCost și StareFinanciara sunt atribute iar idCost și idServiciu sunt chei candidat. |
| Adresa | idAdresa este cheia priamara find nici un alt atribut dintre Oras, Strada și NrCasa sau bombinatii de atribute nu prezintă proprietatea de unicitate. |
| Client | idClient este cheia priamara; nume, prenume, StareFinanciara sunt atribute ce nu pot forma unicitate în tabel iar idAdresa este cheie candidat. |
| Contract | idContract este cheia priamara; InceputContract, SfarsitContract, SumaLunara sunt atribute ce nu prezintă unicitate iar idFactura și idTelefon sunt chei candidat. |
| Cost | idCost este cheia priamara; SMSNat,MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt, ServiciuDateInt sunt atribute ce nu prezintă unicitate și nici combinașiile acestora. |
| Factura | idFactura este cheia priamara; FacturaGenerata, FacturaScadenta și suma sunt atribute care nu prezintă unicitate și nici într-o formă combinată. |
| Prepay | idPrepay este cheia priamara; Credit este singurul atribut ce nu prezinta unicitate iar idCost și idServiciu sunt chei candidat. |
| Serviciu | idServiciu este cheia priamara; Voce, SMS, MMS, ApelVoce, ServiciuDate, Roming sunt atribute ce nu prezinta proprietatea de unicitate. |
| Subscriptie | idSubscriptie este cheia priamara; idClient, idAbonament, idPrepay, idCost, idTelefon și idContract sunt chei candidat. |
| Telefon | idTelefon este cheia priamara, iar NrTelefon este atribut. |

Tabel 2 Cheia priamara

## 3.5 Etapa alegerii SGBD-ului

Alegerea unui SGBD pentru implementarea unei baze de date se face ţinând seama de mai mulţi factori: tehnici, economici şi administrativi. Factorii tehnici trebuie să asigure alegerea unui SGBD adecvat sarcinii de realizare a sistemului informatic al întreprinderii. De asemenea, înainte de achiziţionarea unui anumit SGBD trebuie să se cunoască configuraţia hardware-software necesară, precum şi alte cerinţe de dezvoltare a programelor de aplicaţii (compilatoare, drivere, etc.). Lucrarea se va implementa cu baza de date relaţională Oracle Database 18c Express Edition.

## 3.6 Crearea Tabelelor

### 3.6.1 Crearea și alterarea unei tabele

Pentru a crea o nouă tabelă se folosește comanda CREATE TABLE. Una dintre cele mai simple forme a acestei comenzi este când informația de bază pentru fiecare coloană este definită împreună cu tipul de dată și dimensiunea. Numele coloanelor într-o tabelă trebuie să fie unice.

CREATE TABLE[schema] nume\_tabelă

(nume\_coloană\_1 tip\_1(dimensiune) [DEFAULT expresie],

nume\_coloană\_2 tip\_2(dimensiune) [DEFAULT expresie],

...);

Semnificația termenilor din comandă:

|  |  |
| --- | --- |
| schema | numele posesorului tabelului |
| nume\_tabelă | numele tabelului |
| nume\_coloană | numele coloanei |
| tip | tipul de dată din coloana respectivă |
| DEFAULT expresie | specifică valoarea implicită, dacă aceasta lipsește într-o comandă ulterioară de inserare (INSERT) |

Tabel 3 Semnificații termenilor din comanda de creare

Pentru referirea tabelelor unui alt utilizator: tabelele ale căror proprietar sunt alți utilizatori nu sunt în shema utilizatorului curent iar pentru a fi referite, trebuie folosit numele proprietarului tabelului.

Unei coloane îi poate fi alocată o valoare implicită prin optiunea DEFAULT. Aceasta previne apariția de null-uri (sau erori, daca NOT NULL este specificată) dacă o linie este inserată făra o valoare din coloană. Funcții ca SYSDATE si USER sunt valide.

Această sintaxă nu include constrângeri (de domeniu, de tuplu, de dependența datelor), însă acestea se pot defini la crearea tabelului. Constrângerile sunt de doua tipuri: constrângeri de tabel se pot referi la una sau mai multe coloane și constrângerule de coloană acestea făcând referire la o singură coloană. Toate detaliile despre constrangeri sunt stocate în Dictionarul de Date. O altă metodă pentru impunerea constrângerilor este adăugarea acestora în tabel după ce tabelul a fost creat ( ALTER TABLE ). Cu ajutorul comenzii ALTER TABLE se pot realiza următoarele acțiuni:

* adăugare de coloană utilizând clauza ADD:

ALTER TABLE nume\_tabelă

ADD (nume\_coloană tip [DEFAULT expr]

[, column datatype] …);

**Observație**: Se pot adăuga, modifica coloane, dar nu se pot elimina din tabelă. Nu se poate specifica locul de apariție al noii coloane.

* modificare de coloană existentă utilizând MODIFY

ALTER TABLE nume\_tabelă

MODIFY (nume\_coloană tip [DEFAULT expr]

[, column datatype] …);

**Observație**: Modificările permise pot fi schimbarea tipului de dată, mărimea

și valoarea inițială.

* eliminarea uneia sau mai multor coloane dintr-o tabelă utilizând DROP:

ALTER TABLE nume\_tabelă

DROP COLUMN nume\_coloană

Comenzile sunt folosite dacă se dorește modificarea structurii unei tabele.

|  |  |
| --- | --- |
| nume\_tabelă | numele tabelului |
| nume\_coloană | numele coloanei |
| tip | tipul de dată din coloana respectivă |
| DEFAULT expresie | specifică valoarea implicită, dacă aceasta lipsește într-o comandă ulterioară de inserare (INSERT) |

Tabel 4 Comenzi alterare tabel

* pentru adăugarea cheii primare este folosită următoarea sintaxă:

ALTER TABLE nume\_tabelă

ADD CONTRAINT nume\_contrângere PRIMARY KEY (nume\_coloană\_primară)

|  |  |
| --- | --- |
| nume\_contrângere | reprezintă un numele ales pentru constrângere |
| *RIMARY KEY* | definește cheia primară ce constă în coloana sau mulțimea de coloane ce urmează dupa aceasta; |
| nume\_coloană\_primară | reprezintă coloana sau mulțimea de coloane definită drept cheie primară |

Tabel 5 Comenzi adaugare constrangere

* în cazul adăugării unei chei străine, comanda SQL este următoarea:

ALTER TABLE nume\_tabelă

ADD CONTRAINT nume\_constrângere FOREIGN KEY (coloană\_cheie\_străină) REFERENCES nume\_tabel\_referință (coloană\_cheie\_străină) ON DELETE CASCADE/SET NULL

|  |  |
| --- | --- |
| nume\_constrângere | numele ales pentru constrângere |
| FOREIGN KEY | clauză ce definește cheia străină ce constă în coloana sau mulțimea de coloane ce urmează dupa aceasta |
| Coloană\_cheie\_străină | coloana sau mulțimea de coloane definită drept cheie străină |
| REFERENCES nume\_tabel\_referință (coloană\_cheie\_străină) | indică tabelul și coloana la care se face referire |
| ON DELETE CASCADE | indică faptul că atunci când rândul din tabelul referit este șters, acesta va fi șters automat și din tabelul ce referă |
| ON DELETE SET NULL | indică faptul că atunci când rândul din tabelul referit este șters, se realizează conversia valorii cheii străine dependente în NULL |

Tabel 6 Comenzi adaugare de cheie străină

O cheie străină reprezintă o constrângere referenţială care asigură modelarea corectă a asocierilor 1:N între două sau mai multe relaţii.

### 3.6.2 Crearea tabelei Client

Tabela Clientconține informații referitoare la toți clienții unui operator de telefonie mobilă, informații ce sunt păstrate în baza de date Oracle sub forma unor coloane. Coloanele tabelului Client sunt: idClient, idAdresa, idSubsc, nume, prenume și stareFinanciara. Există 2 chei straine ce fac referire la tabelul Adresa si la tabelul Subscriptie. Constrângerile vor fi adăugate ulteriorpentru cheile străine. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

CREATE TABLE Client (

idClient number(10) NOT NULL,

idAdresa number(10) NOT NULL,

Nume varchar2(20) NOT NULL,

Prenume varchar2(30) NOT NULL,

StareFinanciara VARCHAR2(10) DEFAULT 'la zi' NOT NULL);

ALTER TABLE Client ADD CONSTRAINT primkeyClient PRIMARY KEY (idClient);

Tipurile de date pentru atributele folosite pentru a crea tabela Client sunt varchare2 și number. Numărul din paranteză al tipului de date number specifică dimensiunea câmpului, tip de date folosit pentru id-ul de identificare al clienților, adreselor și subscripțiilor. Tipul de date varchare2 este un șir de lungieme variabilă având numărul maxim de caractere de 400 și minim de un caracter, tip de date ce a fost folosit pentru nume, prenume și stareFinanciară.

Va trebui să adăugăm și cheia primară, lucru ce va fi realizat cu comanda ALTER TABLE adăugându-se astfel constrângerea de cheie primară. Comanda folosită pentru setarea atributului de cheie primară atributului idClient este:

O imagine care conține text

Descriere generată automat

Figura 3. 3 Comanda de creare adăugare a cheii primare pentru tabela Client

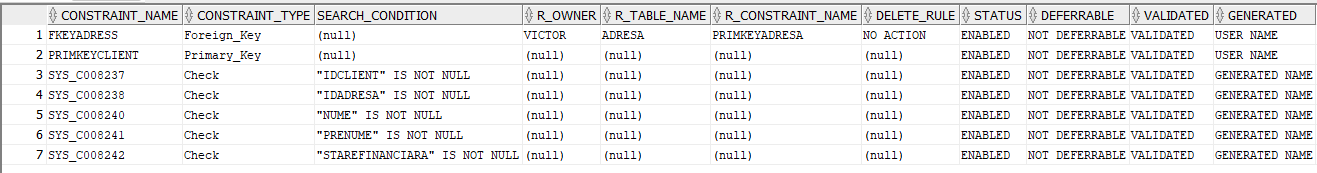


Figura 3. 4 Tabelul Client după creare și setarea cheii primare

### 3.6.3 Crearea tabelei Adresa

Tabela Adresaconține informații referitoare la adresele tuturor cliențiilor unui operator de telefonie mobilă, informații ce sunt păstrate în baza de date Oracle sub forma unor coloane. Coloanele tabelului Adresa sunt: idAdresa, oras, strada și numar. Nu există nici o chei straină. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

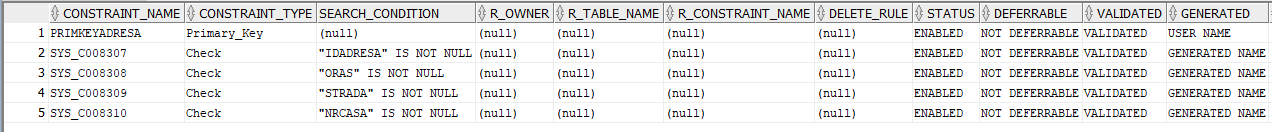


Figura 3. 5 Tabelul Adresă după creare și setarea cheii primare

CREATE TABLE Adresa (

idAdresa number(10) NOT NULL,

Oras varchar2(20) NOT NULL,

Strada varchar2(30) NOT NULL,

NrCasa VARCHAR2(4) NOT NULL);

ALTER TABLE Adresa ADD CONSTRAINT primkeyAdresa PRIMARY KEY (idAdresa)

### 3.6.4 Crearea tabelei Factura

Tabela Factura conține informații referitoare la plățile efectuate de clienți, informații ce sunt păstrate în baza de date Oracle sub forma unor coloane. Coloanele tabelului Factura sunt: idFactura, idContract, suma, facturaGenerata și facturaScadenta. Există o chei straină ce face referire la tabelul Contract. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

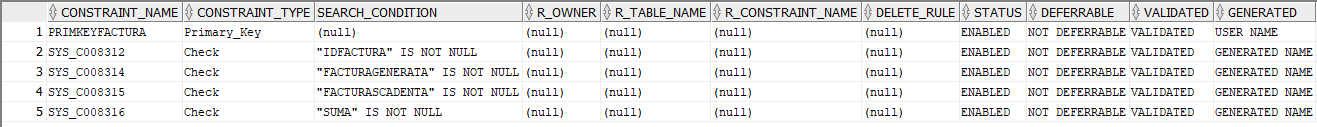


Figura 3. 6 Tabelul Factură după creare și setarea cheii primare

CREATE TABLE Factura(

idFactura number(10) NOT NULL,

idContract number(10) NOT NULL,

FacturaGenerata DATE NOT NULL,

FacturaScadenta DATE NOT NULL,

Suma number(10) NOT NULL,

CONSTRAINT primkeyFactura PRIMARY KEY (idFactura) );

### 3.6.5 Crearea tabelei Contract

Tabela Contract conține informații referitoare la contractele create între operatorul de telefonie mobilă și clienți, informații ce sunt păstrate în baza de date Oracle sub forma unor coloane. Coloanele tabelului Contract sunt: idContract, idFactura, idTelefon, startContract, terminaContract și sumaLunara. Există doua chei straină ce face referire la tabelul Factura pentru a identifica factura pe baza contractului și la tabelul Telefon pentru a identifica numarul de telefon din contract. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

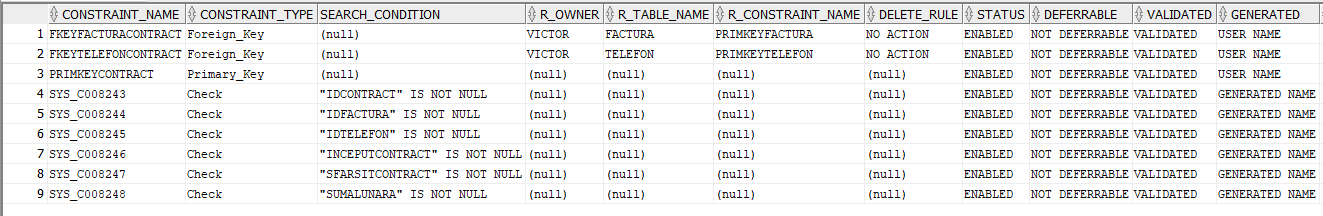


Figura 3. 7 Tabelul Contract după creare și setarea cheii primare

CREATE TABLE Contract(

idContract number(10) NOT NULL,

idFactura number(10) NOT NULL,

idTelefon number(10) NOT NULL,

InceputContract DATE NOT NULL,

SfarsitContract DATE NOT NULL,

SumaLunara number(10) NOT NULL,

CONSTRAINT primkeyContract PRIMARY KEY (idContract) );

### 3.6.6 Crearea tabelei Telefon

Tabela Telefon conține informații referitoare la numerele de telefon care sunt înregistrate de către operatorul de telefonie mobila atât pentru abonamente cât și pentru cartelele pre-pay prin intermediul coloanei idSubsc ce face legătura la tabelul Subscriptie.Coloanele sunt idTelefon, idSubsc și nrTelefon. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

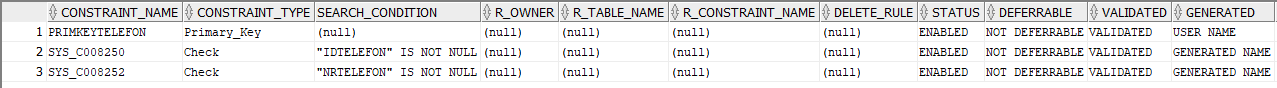


Figura 3. 8 Tabelul Telefon după creare și setarea cheii primare

CREATE TABLE Telefon(

idTelefon number(10) NOT NULL,

NrTelefon number(10) NOT NULL,

CONSTRAINT primkeyTelefon PRIMARY KEY (idTelefon) );

### 3.6.7 Crearea tabelei Serviciu

Tabela Serviciu conține informații referitoare la serviciile oferite de către operatorul de telefonie în rețea sau roming atât pentru cartelele pre-pay cât și pentru abonamente. În acest tabel se regăsesc următoarele coloane: idServiciu, voce, sms, mms, videoCall, internet, roming. Nu există cheie străină. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

O imagine care conține text, computer, captură de ecran

Descriere generată automat

Figura 3. 9 Tabelul Serviciu după creare și setarea cheii primare

CREATE TABLE Serviciu(

idServiciu number(1) NOT NULL,

Voce number(1) NOT NULL,

SMS number(1) NOT NULL,

MMS number(1) NOT NULL,

ApelVoce number(1) NOT NULL,

ServiciuDate number(1) NOT NULL,

Roming number(1) DEFAULT 0 NOT NULL,

CONSTRAINT primkeyServiciu PRIMARY KEY (idServiciu) );

### 3.6.8 Crearea tabelei Prepay

Tabela Prepay conține informații referitoare la cartelele Pre-pay din rețea. Coloanele sale sunt: idPrepay, idServiciu, idCost, idSubsc și credit. Cheile străină din acest tabel sunt idServiciu ce face legătura cu serviciile disponibile și tabela Cost care calculeaza costul. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

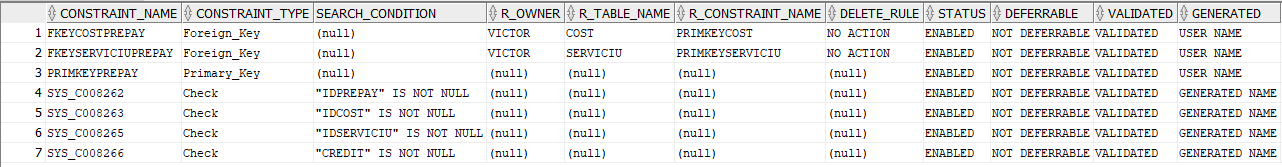


Figura 3. 10 Tabelul Prepay după creare și setarea cheii primare

CREATE TABLE Prepay(

idPrepay number(10) NOT NULL,

idCost number(10) NOT NULL,

idServiciu number(10) NOT NULL,

Credit number(10) NOT NULL,

CONSTRAINT primkeyPrepay PRIMARY KEY (idPrepay) );

### 3.6.9 Crearea tabelei Abonament

Tabela Abonament conține informații referitoare la toți abonații din rețea. Coloanele sale sunt: idAbonament, tipAbonament, idCost, idSubsc, idServiciu, stareFinanciara și extraCost. Cheile străină din acest tabel sunt idServiciu ce face legătura cu serviciile disponibile și tabela Cost care calculeaza costul. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

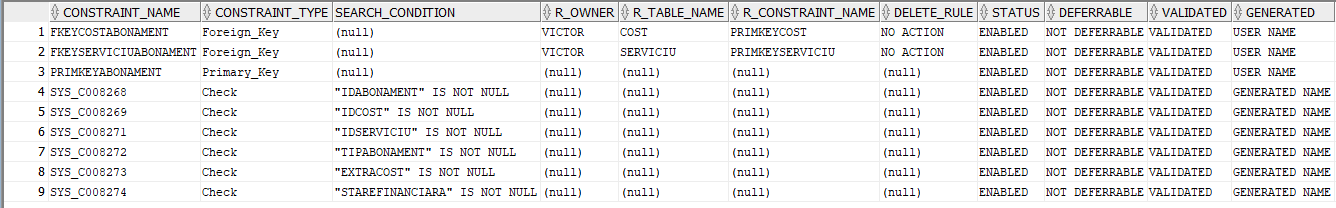


Figura 3. 11 Tabelul Abonament după creare și setarea cheii primare

CREATE TABLE Abonament(

idAbonament number(10) NOT NULL,

idCost number(10) NOT NULL,

idServiciu number(10) NOT NULL,

TipAbonament varchar2(10) NOT NULL,

ExtraCost number(10) NOT NULL,

StareFinanciara number(1) DEFAULT 0 NOT NULL,

CONSTRAINT primkeyAbonament PRIMARY KEY (idAbonament ) );

### 3.6.10 Crearea tabelei Subscriptie

Tabela Subscripție conține informații legate de toată baza de date find puntea de legătură între tabelele principale. Coloanlele ce formează tabela subscripție sunt: idSubscr, idClient, idAbonament, idPrepay, idTelefon, idCost și idContract. Aceasta find singura tabelă care este formată în totalitate din chei străine. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori exceptând coloanele idAbonament și idPrepay acestea putand lua valoarea NULL în momentul în care nu se va face legătura la tabela Abonament respectiv la tabela Prepay .

O imagine care conține text, interior, captură de ecran

Descriere generată automat

Figura 3. 12 Tabelul Subscripție după creare și setarea cheii primare

CREATE TABLE Subscriptie(

idSubsc number(10) NOT NULL,

idClient number(10) NOT NULL,

idAbonament number(10),

idPrepay number(10),

idCost number(10) NOT NULL,

idTelefon number(10) NOT NULL,

idContract number(10) NOT NULL,

CONSTRAINT primkeySubscriptie PRIMARY KEY (idSubsc) );

### 3.6.11 Crearea tabelei Cost

Tabela Cost cuprinde tarifele standard pentru convorbiri în reţea sau în afara reţelei atât pentru abonamente cât şi pentru cartele Prepay. În această tabelă se regăsesc următorele coloane: idCost, idTelefon, smsNat, mmsNat, voceNat, videoCallNat, serviciuDateNat, smsInt, mmsInt, voceInt, videoCallInt și serviciuDateInt. Aceasta find singura tabelă care este formată în totalitate din chei străine. Se va folosi constrângerea NOT NULL pentru toate coloanele tabelului ce va determina ca toate coloanele tabelului să conțină valori.

O imagine care conține masă

Descriere generată automat

Figura 3. 13 Tabelul Cost după creare și setarea cheii primare

CREATE TABLE Cost(

idCost number(10) NOT NULL,

idTelefon number(10) NOT NULL,

SMSNat number(10) NOT NULL,

MMSNat number(10) NOT NULL,

VoceNat number(10) NOT NULL,

VideoCallNat number(10) NOT NULL,

ServiciuDateNat number(10) NOT NULL,

SMSInt number(10) NOT NULL,

MMSInt number(10) NOT NULL,

VoceInt number(10) NOT NULL,

VideoCallInt number(10) NOT NULL,

ServiciuDateInt number(10) NOT NULL,

CONSTRAINT primkeyCost PRIMARY KEY (idCost ) );

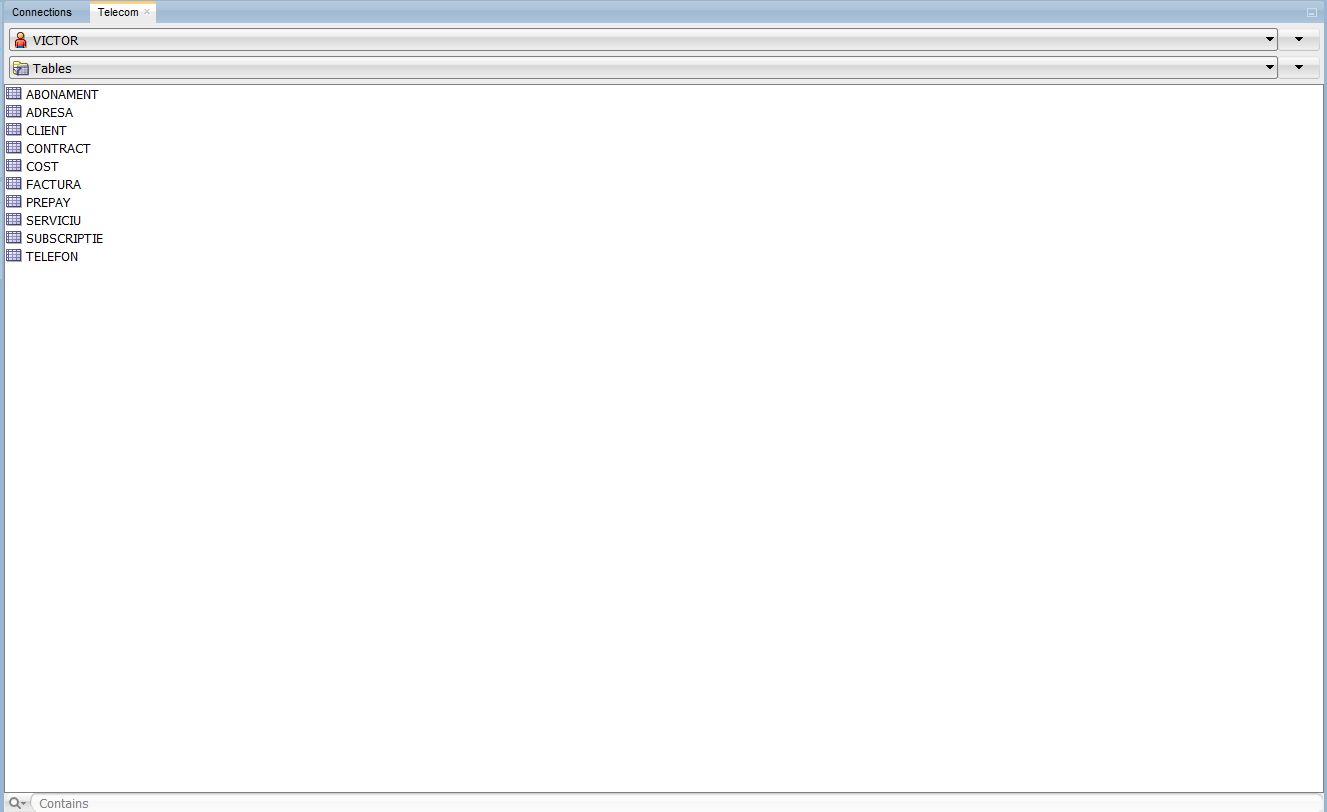


Figura 3. 14 Tabelele create

## 3.7 Introducerea de constrângerilor de tip Foreign Key

### 3.7.1 Introducerea cheii străine idSubsc și a cheii idAdresaîn tabela Client

ALTER TABLE Client ADD CONSTRAINT fKeyAdress FOREIGN KEY (idAdresa) REFERENCES Adresa(idAdresa);

### 3.7.2 Introducerea cheii străine idFactura și cheii idTelefonîn tabela Contract

ALTER TABLE Contract ADD CONSTRAINT fKeyFacturaContract FOREIGN KEY (idFactura) REFERENCES Factura(idFactura);

ALTER TABLE Contract ADD CONSTRAINT fKeyTelefonContract FOREIGN KEY (idTelefon) REFERENCES Telefon(idTelefon);

### 3.7.3 Introducerea cheii străine idServiciu și a cheii idCostîn tabela PrePay

ALTER TABLE PrePay ADD CONSTRAINT fKeyServiciuPrepay FOREIGN KEY (idServiciu) REFERENCES Serviciu(idServiciu);

ALTER TABLE PrePay ADD CONSTRAINT fKeyCostPrepay FOREIGN KEY (idCost) REFERENCES Cost(idCost);

### 3.7.4 Introducerea cheii străine idCost și a cheii idServiciu în tabela Abonament

ALTER TABLE Abonament ADD CONSTRAINT fKeyCostAbonament FOREIGN KEY (idCost) REFERENCES Cost(idCost);

ALTER TABLE Abonament ADD CONSTRAINT fKeyServiciuAbonament FOREIGN KEY (idServiciu) REFERENCES Serviciu(idServiciu);

### 3.7.5 Introducerea cheii străine idTelefon în tabela Cost

ALTER TABLE Cost ADD CONSTRAINT fKeyTELEFONCost FOREIGN KEY (idTelefon) REFERENCES TELEFON(idTelefon);

### 3.7.6 Introducerea cheii străine idClient, idAbonament, idPrepay, idCost, idTelefon și a cheii idContract în tabela Subscriptie

ALTER TABLE Subscriptie ADD CONSTRAINT fKeyClientSubsc FOREIGN KEY (idClient) REFERENCES Client(idClient);

ALTER TABLE Subscriptie ADD CONSTRAINT fKeyAbonamentSubsc FOREIGN KEY (idAbonament) REFERENCES ABONAMENT(idAbonament);

ALTER TABLE Subscriptie ADD CONSTRAINT fKeyPREPAYSubsc FOREIGN KEY (idPrepay) REFERENCES PREPAY(idPrepay);

ALTER TABLE Subscriptie ADD CONSTRAINT fKeyCOSTSubsc FOREIGN KEY (idCost) REFERENCES COST(idCost);

ALTER TABLE Subscriptie ADD CONSTRAINT fKeyTELEFONSubsc FOREIGN KEY (idTelefon) REFERENCES TELEFON(idTelefon);

ALTER TABLE Subscriptie ADD CONSTRAINT fKeyCONTRACTSubsc FOREIGN KEY (idContract) REFERENCES CONTRACT(idContract);

## 3.8 Inspectarea tabelelor prin ”Schemă Browser”

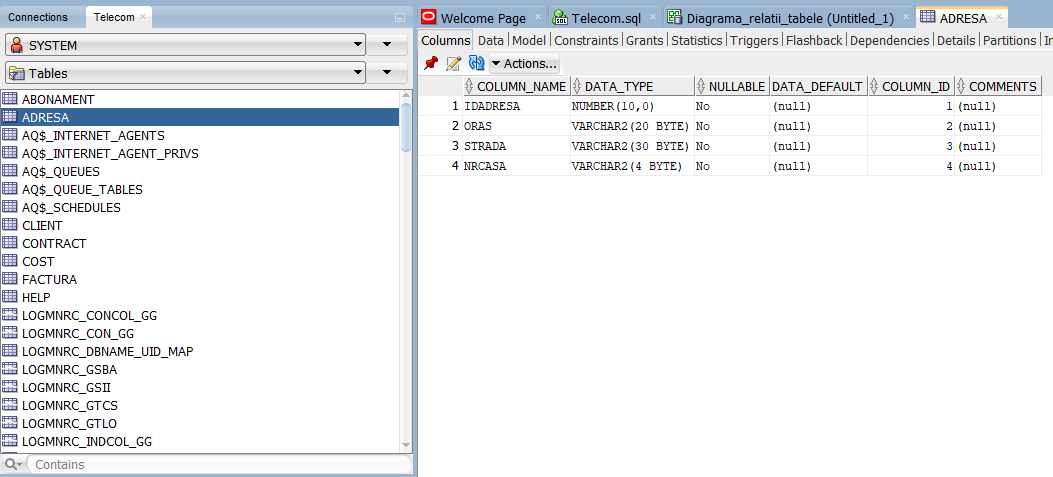


Figura 3. 15 Tabela Adresă

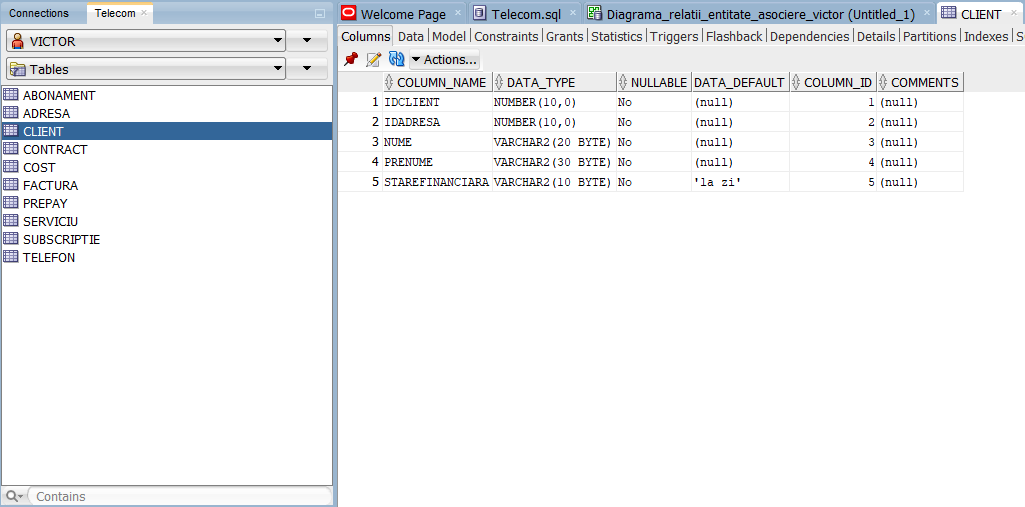


Figura 3. 16 Tabela Client

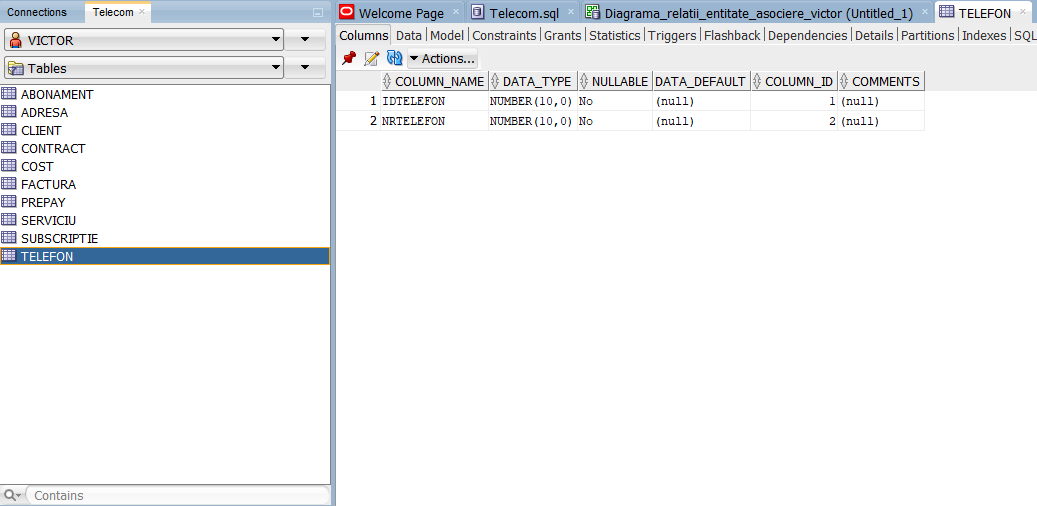


Figura 3. 17 Tabela Telefon

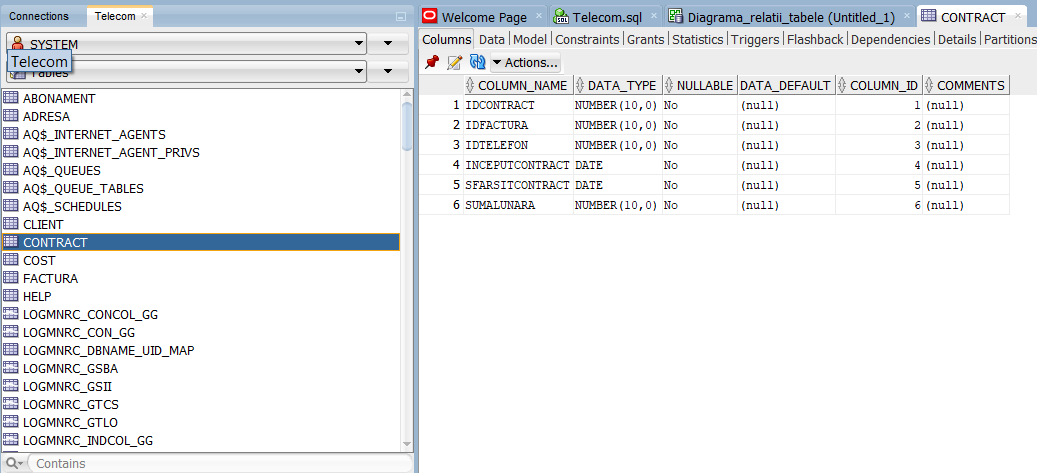


Figura 3. 18 Tabela Contract

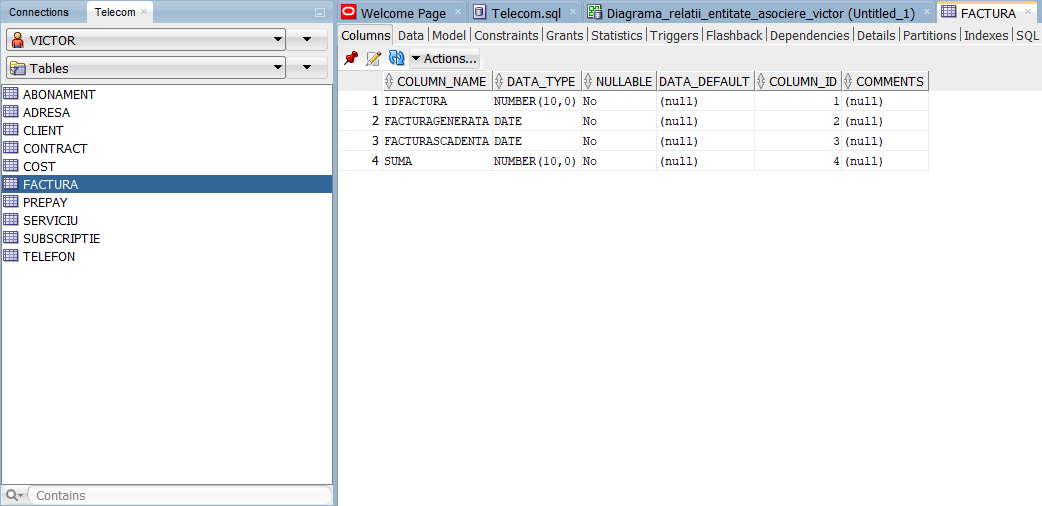


Figura 3. 19 Tabela Factură

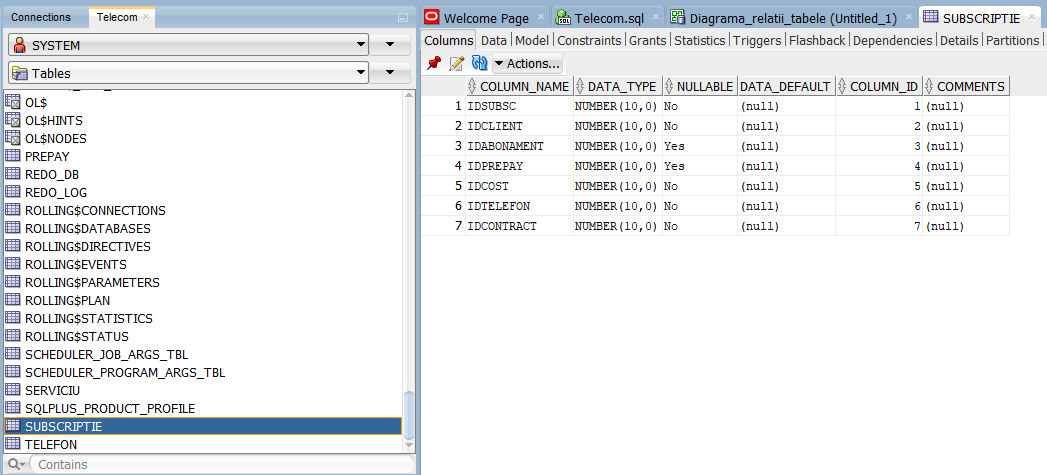


Figura 3. 20 Tabela Subscripție

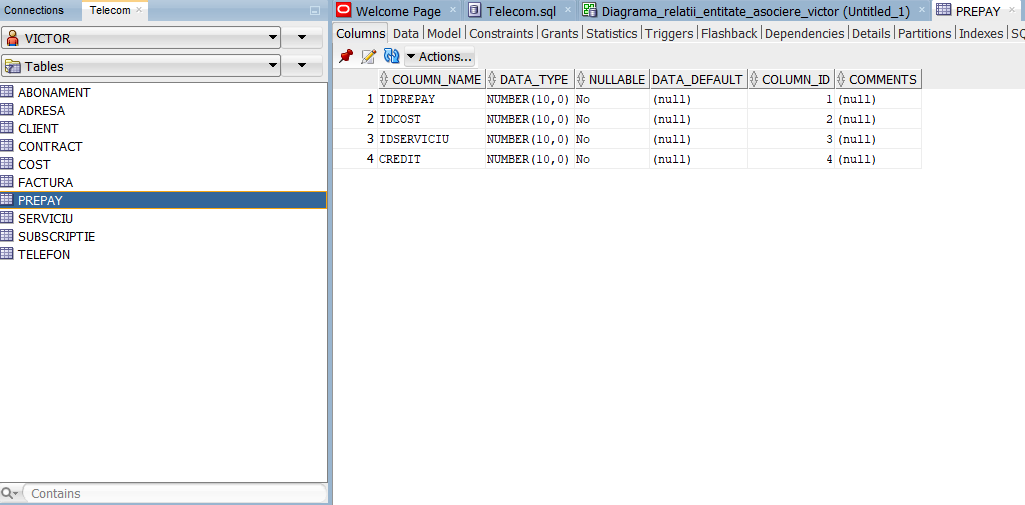


Figura 3. 21 Tabela PrePay

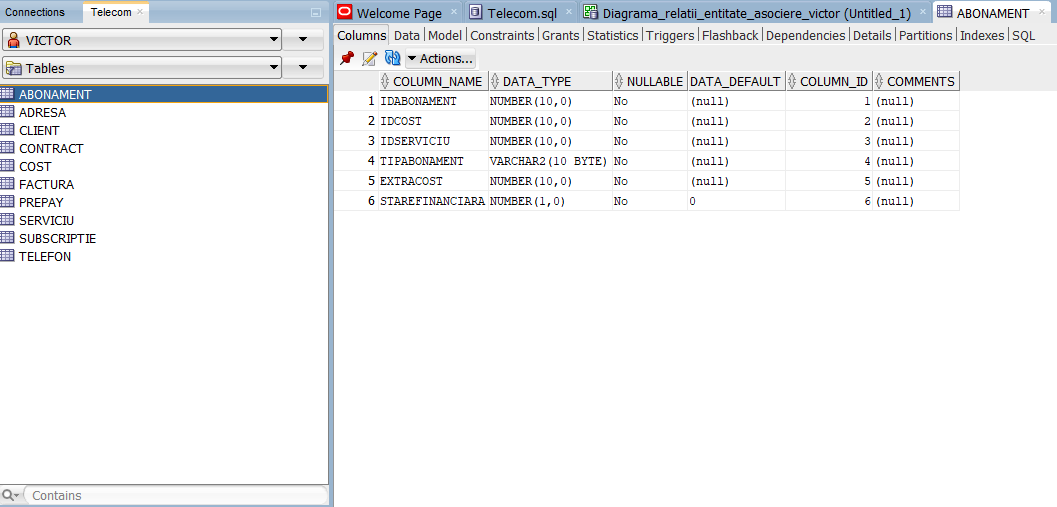


Figura 3. 22 Tabela Abonament

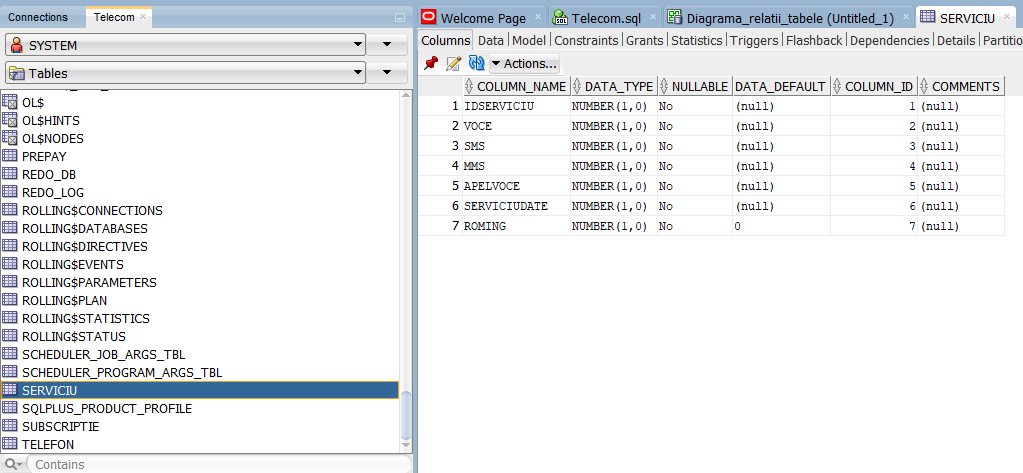


Figura 3. 23 Tabela Serviciu

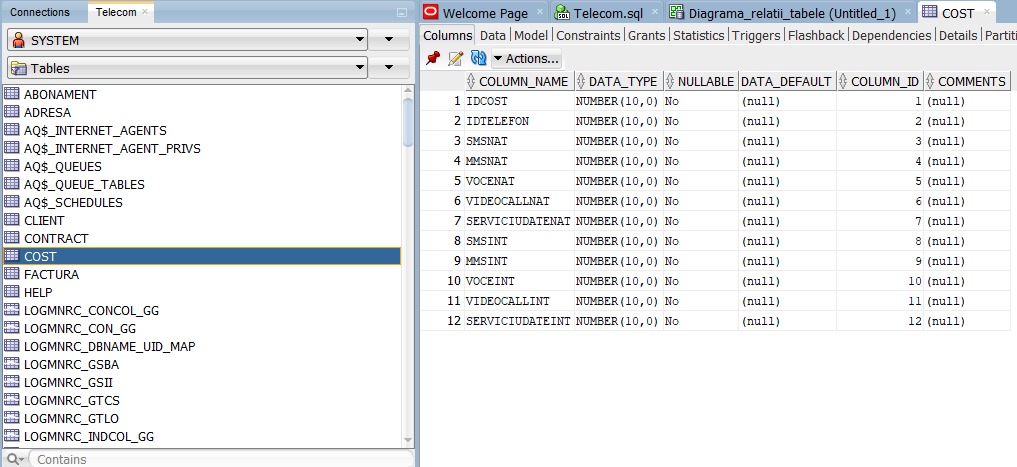


Figura 3. 24 Tabela Cost

## 3.9 Inserarea datelor în tabele

Modificarea tabelei sau vederii prin adăugarea uni rând nou se face prin intermediul comenzii INSERT. Sintaxa comenzii SQL de inserare a datelor într-un tabel este:

***INSERT INTO nume-tabel***

***[(nume-col1,nume-col2,...)]***

***{VALUES (valoare 1,valoare2,...) | cerere );***

Pentru ”*nume-col1, nume-col2...”* precizate în paranteze vor fi furnizate valorile corespunzătoare, iar coloanelor nespecificate le sunt ataşate valori nule.

Coloanele pot fi precizate în orice ordine, însă trebuie asigurată corespondenţa între numele coloanelor şi valorile furnizate.

În cazul în care anumite coloane nu sunt specificate explicit, se impune ca ordinea în care apar valorile în comanda INSERT să coincidă cu cea în care coloanele au fost definite la crearea tabelului.

Dacă nu se mai cunoaşte ordinea de declarare a coloanelor se foloseşte comanda DESCRIBEcare va afişa lista coloanelor definite pentru tabelul respectivă, tipul şi lungimea lor.

Prin forma INSERT...VALUESse introduce în tabel un singur rând. Cu ajutorul valorii NULLse pot introduce valori nule.

Valorile trebuie sa aibă acelaşi tip de dată ca şi câmpurile în care sunt adăugate.

Valorile introduse trebuie să respecte restricţiile de integritate definite la crearea tabelei (de exemplu, câmpuri definite ca NOT NULL sau UNIQUE).

Prin intermediul comenzii SELECT se pot extrage informații din bazele de date. Folosind comanda SELECT, se pot face următoarele acțiuni:

• SELECȚIE (SELECTION): se pot alege liniile de care este nevoie din tabelele de date. Pot fi folosite diferite criterii de selecție, pentru a vizualiza numai ceea ce se dorește, nu un întreg tabel.

• PROIECTARE (PROJECTION): se pot alege coloanele din tabelele de date.

• COMBINAREA (JOIN): se pot uni datele aflate în tabele diferite prin crearea unei legături între coloanele tabelelor de unde provin datele.

***SELECT [ DISTINCT ] {\*, coloană [alias] , …..} FROM tabel ;***

Din punct de vedere sintactic:

**SELECT** este o listă de una sau mai multe coloane;

**DISTINCT** suprimă duplicatele;

**\*** selectează toate coloanele;

**column** numele coloanei;

**alias** dă un alt nume coloanei selectate;

**FROM** table specifică tabela care conține coloanele.

### 3.9.1 Introducerea datelor în tabela Telefon

INSERT ALL

INTO Telefon(idTelefon, NrTelefon) VALUES ('1', '0764839457')

INTO Telefon(idTelefon, NrTelefon) VALUES ('2', '0703940290')

INTO Telefon(idTelefon, NrTelefon) VALUES ('3', '0734342586')

INTO Telefon(idTelefon, NrTelefon) VALUES ('4', '0725657786')

INTO Telefon(idTelefon, NrTelefon) VALUES ('5', '0789757567')

INTO Telefon(idTelefon, NrTelefon) VALUES ('6', '0712435678')

INTO Telefon(idTelefon, NrTelefon) VALUES ('7', '0736565677')

INTO Telefon(idTelefon, NrTelefon) VALUES ('8', '0743243677')

SELECT \*FROM Dual;

Prin intermediul comenzii ”SELECT \* FROM Telefon” se vor putea vizualiza datele introduse anterior în tabelă.

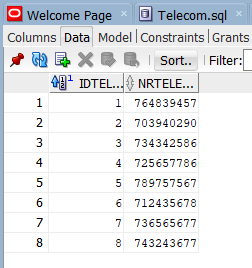


Figura 3. 25 Afișarea datelor în tabela Telefon

### 3.9.2 Introducerea datelor în tabela Factură

INSERT ALL

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('1', TO\_DATE('2020/07/15 16:27:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/08/15 18:27:20', 'yyyy/mm/dd hh24:mi:ss'), '45' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('2', TO\_DATE('2020/08/27 16:35:50', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/09/27 18:35:50', 'yyyy/mm/dd hh24:mi:ss'), '32' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('3', TO\_DATE('2020/05/19 16:17:45', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/06/19 18:17:45', 'yyyy/mm/dd hh24:mi:ss'), '35' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('4', TO\_DATE('2020/06/10 16:19:37', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/07/10 18:19:37', 'yyyy/mm/dd hh24:mi:ss'), '60' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('5', TO\_DATE('2020/09/04 16:08:27', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/10/04 18:08:27', 'yyyy/mm/dd hh24:mi:ss'), '35' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('6', TO\_DATE('2020/07/02 16:45:38', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/08/02 18:45:38', 'yyyy/mm/dd hh24:mi:ss'), '75' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('7', TO\_DATE('2020/06/24 16:57:10', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/07/24 18:57:10', 'yyyy/mm/dd hh24:mi:ss'), '28' )

INTO Factura(idFactura, FacturaGenerata, FacturaScadenta, Suma) VALUES ('8', TO\_DATE('2020/07/08 16:13:02', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/08/08 18:13:02', 'yyyy/mm/dd hh24:mi:ss'), '45' )

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Factura” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține masă

Descriere generată automat

Figura 3. 26 Afișarea datelor din tabela Factură

### 3.9.3 Introducerea datelor în tabela Adresă

INSERT ALL

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('1', 'Craiova', 'Uzinei', '1')

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('2', 'Braila', 'Rosiorilor', '18')

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('3', 'Bucuresti', 'Viorele', '24')

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('4', 'Ploiesti', 'Centrala', '7')

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('5', 'Bucuresti', 'Florilor', '35')

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('6', 'Arad', 'Morarilor', '15')

INTO Adresa (idAdresa, Oras, Strada, NrCasa) VALUES ('7', 'Galati', 'Brailei', '77')

SELECT \* FROM DUAL ;

Prin intermediul comenzii ”SELECT \* FROM Adresa” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține masă

Descriere generată automat

Figura 3. 27 Inserarea datelor în tabela Adresă

### 3.9.4 Introducerea datelor în tabela Contract

INSERT ALL

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('1', '3', '7', TO\_DATE('2018/08/20 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/11/23 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '35')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('2', '2', '6', TO\_DATE('2019/05/15 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/05/15 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '32')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('3', '4', '4', TO\_DATE('2019/12/27 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/12/27 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '60')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('4', '8', '8', TO\_DATE('2018/10/10 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/10/10 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '45')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('5', '7', '2', TO\_DATE('2017/10/13 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2019/10/13 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '28')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('6', '1', '1', TO\_DATE('2018/02/18 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2021/02/18 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '45')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('7', '6', '3', TO\_DATE('2018/07/03 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2021/07/03 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '75')

INTO Contract(idContract, idFactura, idTelefon, InceputContract, SfarsitContract, SumaLunara) VALUES ('8', '5', '5', TO\_DATE('2019/06/26 14:35:20', 'yyyy/mm/dd hh24:mi:ss'), TO\_DATE('2020/06/26 18:25:19', 'yyyy/mm/dd hh24:mi:ss'), '35')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Contract” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține masă

Descriere generată automat

Figura 3. 28 Afișarea datelor din tabela Contract

### 3.9.5 Introducerea datelor în tabela Cost

INSERT ALL

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('1', '2', '6', '2', '10', '15', '17', '13', '21', '142', '42', '12')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('2', '4', '4', '5', '10', '12', '13', '16', '20', '12', '12', '22')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('3', '3', '2', '7', '10', '15', '11', '13', '30', '52', '32', '51')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('4', '8', '3', '5', '13', '21', '22', '15', '22', '24', '21', '42')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('5', '5', '7', '8', '15', '22', '30', '18', '25', '19', '34', '45')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('6', '7', '2', '3', '15', '20', '20', '10', '24', '20', '25', '45')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('7', '6', '4', '6', '15', '20', '25', '35', '30', '27', '40', '50')

INTO Cost(idCost, idTelefon, SMSNat, MMSNat, VoceNat, VideoCallNat, ServiciuDateNat, SMSInt, MMSInt, VoceInt, VideoCallInt,

ServiciuDateInt) VALUES ('8', '1', '8', '9', '17', '25', '10', '20', '16', '30', '27', '50')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Cost” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține text, interior, captură de ecran

Descriere generată automat

Figura 3. 29 Afișarea datelor din tabela Cost

### 3.9.6 Introducerea datelor în tabela Serviciu

INSERT ALL

INTO Serviciu (idServiciu, Voce, SMS, MMS, ApelVoce, ServiciuDate ) VALUES ('1', '1', '1', '0', '1', '1')

INTO Serviciu (idServiciu, Voce, SMS, MMS, ApelVoce, ServiciuDate, Roming) VALUES ('2', '1', '1', '1', '0', '0', '0')

INTO Serviciu (idServiciu, Voce, SMS, MMS, ApelVoce, ServiciuDate, Roming) VALUES ('3', '1', '1', '1', '0', '1', '1')

INTO Serviciu (idServiciu, Voce, SMS, MMS, ApelVoce, ServiciuDate ) VALUES ('4', '1', '0', '0', '0', '0')

INTO Serviciu (idServiciu, Voce, SMS, MMS, ApelVoce, ServiciuDate, Roming) VALUES ('5', '1', '1', '0', '0', '1', '0')

INTO Serviciu (idServiciu, Voce, SMS, MMS, ApelVoce, ServiciuDate, Roming) VALUES ('6', '1', '1', '1', '1', '1', '0')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Serviciu” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține masă

Descriere generată automat

Figura 3. 30 Afișarea datelor din tabela Serviciu

### 3.9.7 Introducerea datelor în tabela PrePay

INSERT ALL

INTO Prepay(idPrepay, idCost, idServiciu, Credit) VALUES ('1', '1', '4', '3')

INTO Prepay(idPrepay, idCost, idServiciu, Credit) VALUES ('2', '2', '6', '15')

INTO Prepay(idPrepay, idCost, idServiciu, Credit) VALUES ('3', '4', '2', '20')

INTO Prepay(idPrepay, idCost, idServiciu, Credit) VALUES ('4', '6', '5', '16')

INTO Prepay(idPrepay, idCost, idServiciu, Credit) VALUES ('5', '3', '3', '30')

INTO Prepay(idPrepay, idCost, idServiciu, Credit) VALUES ('6', '5', '1', '40')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Prepay” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține masă

Descriere generată automat

Figura 3. 31 Afișarea datelor din tabela PrePay

### 3.9.8 Introducerea datelor în tabela Abonament

INSERT ALL

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost, StareFinanciara) VALUES ('1', '3', '4', 'ExtraXXL', '10', '1')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost, StareFinanciara) VALUES ('2', '5', '2', 'NetPlus', '5', '1')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost) VALUES ('3', '8', '5', 'Exclusiv', '5')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost, StareFinanciara) VALUES ('4', '1', '4', 'Infinity', '3', '1')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost, StareFinanciara) VALUES ('5', '2', '3', 'Red 10', '4', '1')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost) VALUES ('6', '4', '1', 'Red 14', '2')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost, StareFinanciara) VALUES ('7', '6', '6', 'InterNet', '6', '1')

INTO Abonament(idAbonament, idCost, idServiciu, TipAbonament, ExtraCost) VALUES ('8', '7', '1', 'Red 20', '15')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Abonament” se vor putea vizualiza datele introduse anterior în tabelă.

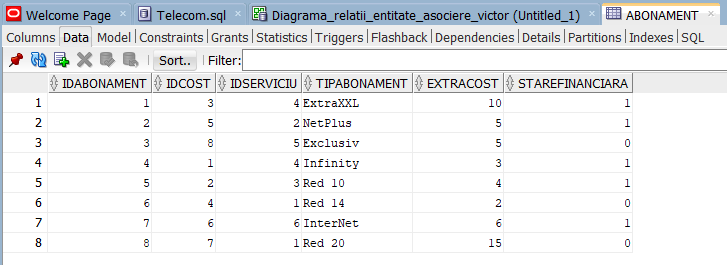


Figura 3. 32 Afișarea datelor din tabela Abonament

### 3.9.9 Introducerea datelor în tabela Client

INSERT ALL

INTO Client (idClient, idAdresa, Nume, Prenume) VALUES ('1', '1', 'Teodosiade', 'Victor')

INTO Client (idClient, idAdresa, Nume, Prenume, StareFinanciara) VALUES ('2', '2', 'Stanica', 'Marius', 'intarziat')

INTO Client (idClient, idAdresa, Nume, Prenume) VALUES ('3', '3', 'Dumitriu', 'Paul')

INTO Client (idClient, idAdresa, Nume, Prenume, StareFinanciara) VALUES ('4', '4', 'Enescu', 'Gabriela', 'intarziat')

INTO Client (idClient, idAdresa, Nume, Prenume) VALUES ('5', '5', 'Bontea', 'Sorin')

INTO Client (idClient, idAdresa, Nume, Prenume, StareFinanciara) VALUES ('6', '6', 'Scarlatescu', 'Catalin', 'intarziat')

INTO Client (idClient, idAdresa, Nume, Prenume) VALUES ('7', '7', 'Dumitrescu', 'Florin')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Client” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține masă

Descriere generată automat

Figura 3. 33 Afișarea datelor din tabela Client

### 3.9.10 Introducerea datelor în tabela Subscripție

INSERT ALL

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('1', '5', '1', '2', '4', '1', '1')

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('2', '4', '2', '4', '5', '2', '2')

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('3', '2', '3', '3', '1', '5', '3')

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('4', '6', '4', '5', '2', '6', '5')

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('5', '1', '5', '7', '7', '4', '6')

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('6', '7', '6', '6', '8', '', '7')

INTO Subscriptie (idSubsc, idClient, idContract, idTelefon, idAbonament, idPrepay, idCost) VALUES ('7', '5', '7', '1', '3', '', '8')

SELECT \* FROM DUAL;

Prin intermediul comenzii ”SELECT \* FROM Subscriptie” se vor putea vizualiza datele introduse anterior în tabelă.

O imagine care conține text, interior, captură de ecran

Descriere generată automat

Figura 3. 34 Afișarea datelor din tabela Subscripție

## 3.10 Creare Vederi

O vedere reprezintă o relaţie virtuală care nu este de sine stătătoare, ci este derivată în mod dinamic din una sau mai multe relaţii de bază. O vedere are aspectul unui tabel conținând coloane și linii care pot fi actualizate și în care se pot efectua inserări sau eliminări. O vedere este de fapt un tabel logic care nu stochează date. Ea își preia datele din tabelele sau vederile pe care se bazează. Toate operațiile efectuate asupra unei vederi afectează practic tabelele de bază ale vederii.

O vedere este creată folosind o interogare şi prin urmare poate fi privită ca fiind un tabel virtual sau o interogare stocată.Vederile sunt dinamice și afișează întotdeauna informațiile curente ale tabelelor. Atunci când tabelele vederii sunt manipulate, aceste modificari sunt reflectate instantaneu în vedere. Vederile permit afişarea datelor într-o formă diferită de cea în care sunt stocate în tabele. Vederile permit adaptarea prezentării datelor în conformitate cu cerinţele specifice ale diverselor tipuri de utilizatori.

În general, vederile sunt create în urmatoarele scopuri:

• Asigurarea unui nivel mai înalt de securitate al bazei de date prin restrângerea accesului la un număr predeterminat de coloane şi linii ale unui tabel. Acest lucru permite utilizatorilor sa vadă un subset restrâns al datelor;

• Simplificarea prezentării datelor prin ascunderea structurilor reuniunilor şi tabelelor care stau la baza vederii;

• Afişarea datelor într-o altă reprezentare decât cea a tabelelor de bază.

Pentru a crea o vedere, se foloseşte comanda CREATE, o vedere se poate defini folosind orice interogare care face referire la tabele sau alte vederi. Sintaxa pentru crearea unei vederi este următoarea:

***CREATE VIEW nume\_vedere AS SELECT coloana1, coloana2, coloana3,..., coloanai***

***FROM nume\_tabel1, nume\_tabel2, ..., nume\_tabeli***

***WHERE condiția1 AND condiția2 AND ...***

• CREATE VIEW nume\_vedere este comanda de creare a vederii cu numele nume\_vedere;

• Cuvântul cheie AS face referire la modul în care vederea va fi contruită; 61

• SELECT coloana1, coloana2, coloana3,..., coloanai este comanda de selcţie a coloanelor care se doresc a face parte din vedere. Aceste coloane pot aparţine mai multor tabele. În cazul în care două coloane ce fac parte din tabele diferite au acelaşi nume, acestea vor fi diefernţiate prin adăugarea numelui tabelului in fata numelui coloanei. Spre exemplu nume\_tabel1.coloana;

• FROM nume\_tabel1, nume\_tabel2, ..., nume\_tabeli este comanda prin care se identifică tabelele care vor contribui la crearea vederii;

• WHERE reprezintă condiţia care se pune asupra înregistrărilor tabelei. Pentru a modifica definitia unei vederi, vederea trebuie înlocuită. Vederile pot fi înlocuite în doua moduri:

• Vederea poate fi distrusă şi apoi recreată cu noua definiţie;

• Vederea poate fi recreatăprin redefinirea ei cu instrucţiunea CREATE VIEW cu clauza opţională OR REPLACE. Această metodă este folosită pentru înlocuirea definiţiei curente a unei vederi.

Vederile sunt tratate similar cu tratarea tabelelor în instructiunile SQL. Adică, pentru a interoga o vedere se foloseşete tot instrucţiunea SELECT:

***SELECT coloana1, coloana2 FROM nume\_vedere.***

Pentru a insera o valoare în vedere se foloseşte instrcţiunea INSERT, ca şi la tabele.

***INSERT INTO nume\_vedere VALUES (...)***

Vederile se caracterizează printr-o serie de limitări:

• Nu se poate folosi o vedere pentru a efectua operatii de inserare , actualizare sau ştergere atunci când interogarea vederii conţine o operatie JOIN, operatorii SET sau DISTINCT, o clauza GROUP BY sau o functie GROUP.

• Nu se poate utiliza o vedere definită cu clauza WITH CHECK OPTION pentru a insera sau a actualiza tabelele de bază.

• Nu pot fi inserate linii în tabele utilizând o vedere care a fost creată folosind expresia DECODE.

Pentru a distruge o vedere, se utilizează comanda DROP:

#### ***DROP VIEW***

### 3.10.1 Crearea vederii de Roaming

Pentru a putea vedea clienții ce au opțiunea de roming activată au fost create două vederi separate cu numele: Client\_Roming\_PREPAY și Client\_Roming\_ABONAMENT.

Pentru crearea acestor vederi s-au folosit o joncţiuni naturale, simple și compuse între tabelele existente în baza de dată. O joncţine naturală este o relaţie care se obţine în felul următor: se calculează produsul cartezian al celor doua relatii, din tuplurile produsului cartezian se selecteza acele tupluri care au valori egale pentru atributele comune şi apoi se face proiecţia rezultatului pe mulţimea de atribute corespunzătoare tabelelor participante. Produsul cartezian dintre două relaţii reprezintă concatenarea valorilor atributelor fiecărui tuplu din prima relaţie cu valorile atributelor tuturor tuplurilor din a doua relaţie.

Crearea vederii pentru clientii PrePay:

CREATE VIEW Clienti\_Roaming\_PREPAY AS

SELECT Client.idClient, Client.Nume, Client.Prenume, Telefon.NrTelefon, Serviciu.idServiciu, Serviciu.Roming

FROM Client, Serviciu, Subscriptie, Prepay, Telefon

WHERE Client.idClient=Subscriptie.idClient AND (Subscriptie.idTelefon=Telefon.idTelefon AND Subscriptie.idPrepay=Prepay.idPrepay) AND Prepay.idServiciu=Serviciu.idServiciu AND Serviciu.Roming=1;

Rezultatul primei vederi se poate afișa prin comanda ”SELECT \* FROM Clienti\_Roming\_PREPAY” sau prin acesarea ”Object Browser”>” Clienti\_Roming\_PREPAY”>”Date”.

O imagine care conține text, masă

Descriere generată automat

Figura 3. Afișarea tabelei Clienti\_Roaming\_PREPAY

Joncțiunea a fost realizată între tabelele: CLIENT, SERVICIU, SUBSCRIPTIE, PREPAY, TELEFON, folosind ca atribute datele de identificare ale clientului, subscriptiei, telefonului, prepay și serviciului.

Crearea vederii pentru clientii ce prezintă un abonament:

CREATE VIEW Clienti\_Roaming\_ABONAMENT AS

SELECT Client.idClient, Client.Nume, Client.Prenume, Telefon.NrTelefon, Serviciu.idServiciu, Serviciu.Roming

FROM Client, Serviciu, Subscriptie, Abonament, Telefon

WHERE Client.idClient=Subscriptie.idClient AND (Subscriptie.idTelefon=Telefon.idTelefon AND Subscriptie.idAbonament=Abonament.idAbonament) AND Abonament.idServiciu=Serviciu.idServiciu AND Serviciu.Roming=1;

Rezultatul primei vederi se poate afișa prin comanda ”SELECT \* FROM Clienti\_Roming\_ABONAMENT” sau prin acesarea ”Object Browser”>” Clienti\_Roming\_ ABONAMENT”>”Date”.

O imagine care conține text

Descriere generată automat

Figura 3. 36 Afișarea tabelei Clienti\_Roaming\_ABONAMENT

Joncțiunea a fost realizată între tabelele: CLIENT, SERVICIU, SUBSCRIPTIE, ABONAMENT, TELEFON, folosind ca atribute datele de identificare ale clientului, subscriptiei, telefonului, abonamentului și serviciului.

### 3.10.2 Crearea vederii cu clienții care au platit peste o anumita suma de bani

S-a ales ca sumă reprezentatica să fie 35. Pentru a putea spune care este clientul care are de plată mai mult de 35 vom trebui să verificăm toate facturile din tabela facturi după atributul sumă.

CREATE VIEW FacturaPeste35 AS

SELECT Client.Nume, Client.Prenume, Factura.FacturaGenerata, Factura.FacturaScadenta, Factura.Suma

FROM Client, Subscriptie, Contract, Factura

WHERE Client.idClient=Subscriptie.idClient AND Subscriptie.idContract=Contract.idContract AND Contract.idFactura=Factura.idFactura AND Factura.Suma>35

Rezultatul primei vederi se poate afișa prin comanda ”SELECT \* FROM FacturaPeste35” sau prin acesarea ”Object Browser”> ”FacturaPeste35”> ”Date”.

O imagine care conține masă

Descriere generată automat

Figura 3. 37 Afișarea tabelei FacturaPeste35

În urma execuției se va crea o nouă tabelă care v-a afișa: numele, prenumele, data generarii facturii, data scadentă a facturii și suma pe care o are de plătit.

### 3.10.3 Crearea vederii vechime pentru Abonați

Pentru a vedea abonații ce au o anumită vechime în contract s-a ales pentru aces exemplu abonații cu vechime strict mai mare de 2 ani. Pentru a realiza acest lucru a fost folosita comanda:

CREATE VIEW VechimeDe2Ani AS

SELECT Contract.idContract, Client.Nume, Client.Prenume, Telefon.NrTelefon, Contract.InceputContract, Contract.SfarsitContract,

trunc(to\_number(to\_char(trunc(sysdate)-Contract.InceputContract,999999))/365)year

FROM Contract, Client, Telefon, Subscriptie

WHERE Client.idClient=Subscriptie.idClient AND (Subscriptie.idTelefon=Telefon.idTelefon AND Subscriptie.idContract=Contract.idContract) AND trunc(to\_number(to\_char(trunc(sysdate)-Contract.InceputContract,999999))/365) >2

Rezultatul primei vederi se poate afișa prin comanda ”SELECT \* FROM VechimeDe2Ani” sau prin acesarea ”Object Browser”> ” VechimeDe2Ani”> ”Date”.

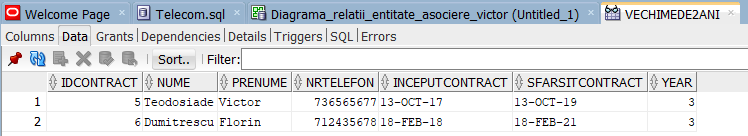


Figura 3. 38 Afișarea tabelei VechimeDe2Ani

# BIBLIOGRAFIE

* **Note de Curs “ Baze de Date pentru Telecomunicatii “ – Dan Galațchi**
* **Baze de date relaţionale şi aplicaţii – Felicia Ionescu, Editura Tehnică, 2004**
* [**https://www.bestjobs.eu/ro/**](https://www.bestjobs.eu/ro/)
* [**https://www.hipo.ro/**](https://www.hipo.ro/)
* ***HTML - The language for building web pages documentation* Accesibil la:** [**https://www.w3schools.com**](https://www.w3schools.com)
* [**https://techterms.com/definition/aspnet**](https://techterms.com/definition/aspnet)
* [**https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/**](https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/)
* [**http://www.anfp.gov.ro/R/Doc/2020/Proiecte/Ghiduri/Ghidul%20Intervievatorului,%202012.pdf**](http://www.anfp.gov.ro/R/Doc/2020/Proiecte/Ghiduri/Ghidul%20Intervievatorului,%202012.pdf)
* [**https://resumelab.com/job-search/hr-statistics?gclid=Cj0KCQjw0Mb3BRCaARIsAPSNGpV3SU\_nuXakrTHKC6UHLKqg3Lj4Hnfh9D1PkDqgcLAEbImY8zr-n5kaAvMjEALw\_wcB**](https://resumelab.com/job-search/hr-statistics?gclid=Cj0KCQjw0Mb3BRCaARIsAPSNGpV3SU_nuXakrTHKC6UHLKqg3Lj4Hnfh9D1PkDqgcLAEbImY8zr-n5kaAvMjEALw_wcB)
* **Teach Yourself SQL in 21 Days, Second Edition**

Tabel Figuri

[Figura 1. 1 Componentele unui sistem de baze de date 6](#_Toc73827813)

[Figura 1. 2 Arhitectura unui sistem de baze de date 7](#_Toc73827814)

[Figura 1. 3 Clasificarea sistemelor de baze de date 8](#_Toc73827815)

[Figura 1. 4 Asocieri unu la unu și unu la multe 10](#_Toc73827816)

[Figura 1. 5 Asocieri multe la unu și multe la multe 10](#_Toc73827817)

[Figura 1. 6 Diagrama entitate-asociere 11](#_Toc73827818)

[Figura 1. 7 Exemplu de diagramă entitate-asociere 12](#_Toc73827819)

[Figura 1. 8 Modelul entitate-asociere extins 13](#_Toc73827820)

[Figura 1. 9 Modelul ierarhic 13](#_Toc73827821)

[Figura 1. 10 Modelul rețea 14](#_Toc73827822)

[Figura 1. 11 Grafic cu complexitatea datelor și a interogărilor 16](#_Toc73827823)

[Figura 1. 12 Sisteme de gestiune a bazelor de date 17](#_Toc73827824)

[Figura 2. 1 Model de tabel creat 19](#_Toc73828243)

[Figura 2. 2 Operatori logici 21](#_Toc73828244)

[Figura 2. 3 Structura comenzilor in Oracle 23](#_Toc73828245)

[Figura 3. 1 Diagrama Entitate-Asociere tip1 24](#_Toc74084901)

[Figura 3. 2 Diagrama Entitate-Asociere tip2 25](#_Toc74084902)

[Figura 3. 3 Comanda de creare adăugare a cheii primare pentru tabela Client 34](#_Toc74084903)

[Figura 3. 4 Tabelul Client după creare și setarea cheii primare 34](#_Toc74084904)

[Figura 3. 5 Tabelul Adresă după creare și setarea cheii primare 34](#_Toc74084905)

[Figura 3. 6 Tabelul Factură după creare și setarea cheii primare 35](#_Toc74084906)

[Figura 3. 7 Tabelul Contract după creare și setarea cheii primare 35](#_Toc74084907)

[Figura 3. 8 Tabelul Telefon după creare și setarea cheii primare 36](#_Toc74084908)

[Figura 3. 9 Tabelul Serviciu după creare și setarea cheii primare 36](#_Toc74084909)

[Figura 3. 10 Tabelul Prepay după creare și setarea cheii primare 37](#_Toc74084910)

[Figura 3. 11 Tabelul Abonament după creare și setarea cheii primare 38](#_Toc74084911)

[Figura 3. 12 Tabelul Subscripție după creare și setarea cheii primare 38](#_Toc74084912)

[Figura 3. 13 Tabelul Cost după creare și setarea cheii primare 39](#_Toc74084913)

[Figura 3. 14 Tabelele create 40](#_Toc74084914)

[Figura 3. 15 Tabela Adresă 42](#_Toc74084915)

[Figura 3. 16 Tabela Client 42](#_Toc74084916)

[Figura 3. 17 Tabela Telefon 43](#_Toc74084917)

[Figura 3. 18 Tabela Contract 43](#_Toc74084918)

[Figura 3. 19 Tabela Factură 44](#_Toc74084919)

[Figura 3. 20 Tabela Subscripție 44](#_Toc74084920)

[Figura 3. 21 Tabela PrePay 45](#_Toc74084921)

[Figura 3. 22 Tabela Abonament 45](#_Toc74084922)

[Figura 3. 23 Tabela Serviciu 46](#_Toc74084923)

[Figura 3. 24 Tabela Cost 46](#_Toc74084924)

[Figura 3. 25 Afișarea datelor în tabela Telefon 48](#_Toc74084925)

[Figura 3. 26 Afișarea datelor din tabela Factură 49](#_Toc74084926)

[Figura 3. 27 Inserarea datelor în tabela Adresă 50](#_Toc74084927)

[Figura 3. 28 Afișarea datelor din tabela Contract 51](#_Toc74084928)

[Figura 3. 29 Afișarea datelor din tabela Cost 52](#_Toc74084929)

[Figura 3. 30 Afișarea datelor din tabela Serviciu 52](#_Toc74084930)

[Figura 3. 31 Afișarea datelor din tabela PrePay 53](#_Toc74084931)

[Figura 3. 32 Afișarea datelor din tabela Abonament 54](#_Toc74084932)

[Figura 3. 33 Afișarea datelor din tabela Client 54](#_Toc74084933)

[Figura 3. 34 Afișarea datelor din tabela Subscripție 55](#_Toc74084934)

[Figura 3. 35 Afișarea tabelei Clienti\_Roaming\_PREPAY 57](#_Toc74084935)

[Figura 3. 36 Afișarea tabelei Clienti\_Roaming\_ABONAMENT 58](#_Toc74084936)

[Figura 3. 37 Afișarea tabelei FacturaPeste35 58](#_Toc74084937)

[Figura 3. 38 Afișarea tabelei VechimeDe2Ani 59](#_Toc74084938)